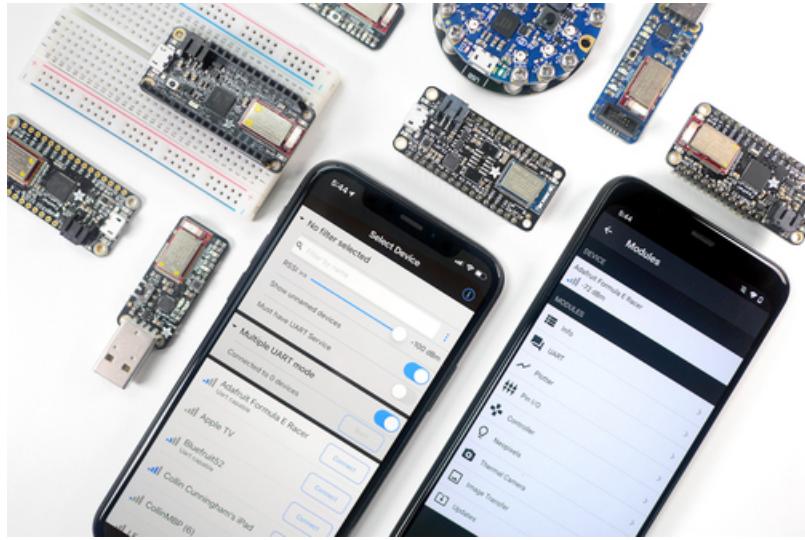




Bluefruit LE Connect for iOS and Android

Created by Collin Cunningham



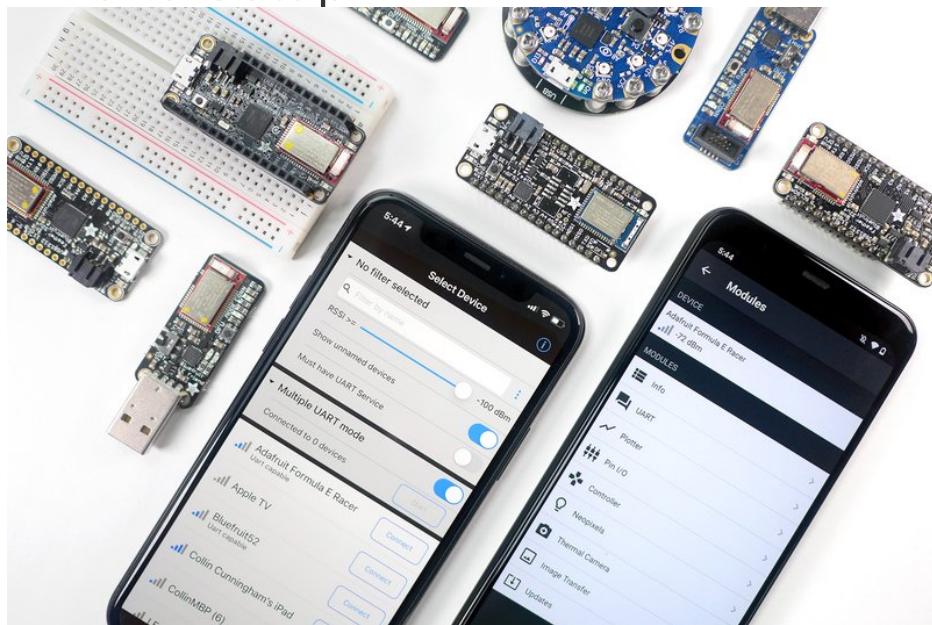
Last updated on 2021-01-04 03:49:13 PM EST

Guide Contents

Guide Contents	2
Installation and Setup	4
Enable Bluetooth	4
Enable Location Services	5
Scan for Devices	6
iPhone	6
iPad	7
Multiple UART Mode	8
Connect	9
Troubleshooting	9
Features	11
Available Modes:	11
Info	13
UART Terminal	14
Main Window	14
Echo	14
ASCII / HEX	15
Copy & Clear	16
Sending Strings	16
Multiple UART Mode	17
MQTT	19
Configuration	19
Usage	22
Received Data	23
Written Data	23
Output Data	23
Plotter	25
Main Plotter View	25
Formatting	27
AutoScroll and Plot Width	29
Controller	31
Format for Sent Data	31
Checksum	31
Sensors	32
Control Pad	33
Color Picker	34
AHRS/Calibration	36
Process	36
Testing	38
Updates	39
Updating Firmware	39
Custom Firmware	40
Image Transfer	42
Data Format	42

Choose Image	42
Bit Depth	43
Resolution	43
Transfer Mode	44
Rotate & Send	46
Pin I/O	47
Wiring Options	49
Arduino with Bluefruit LE Shield	49
Bluefruit Micro or Feather 32u4 Bluefruit	49
Feather M0 Bluefruit LE	49
Bluefruit LE SPI Friend	50
Bluefruit LE UART Friend	50
Flora BLE	50
Library and Config	51
Before loading Firmata BLE...	51
Install Libraries	51
Open Sketch and Configure	51
Bluefruit LE Config	51
Firmata Debug Config	52
Available Pins Config	53
Upload and test	53
Usage	56
Initial Query	56
Digital Input	57
Digital Output	58
PWM Output	59
Analog Input	60
Troubleshooting	62
Cycle Bluetooth	62
Cycle Power	62
Clear Bluetooth Cache	62

Installation and Setup



The Bluefruit LE Connect app provides iOS & Android devices with a variety of tools to communicate with Bluefruit LE devices. These tools cover basic communication and info reporting as well as more project specific uses such as Arduino Pin Control and a Color Picker.

The iOS app is a [free download from Apple's App Store](https://adafru.it/ddu) (<https://adafru.it/ddu>). It requires iOS 11.3 or later and works on iPhones, iPads, and iPod Touches.

The Android app is a free download from the [Google Play Store](https://adafru.it/f4G) (<https://adafru.it/f4G>). It requires Android 4.4 or later.

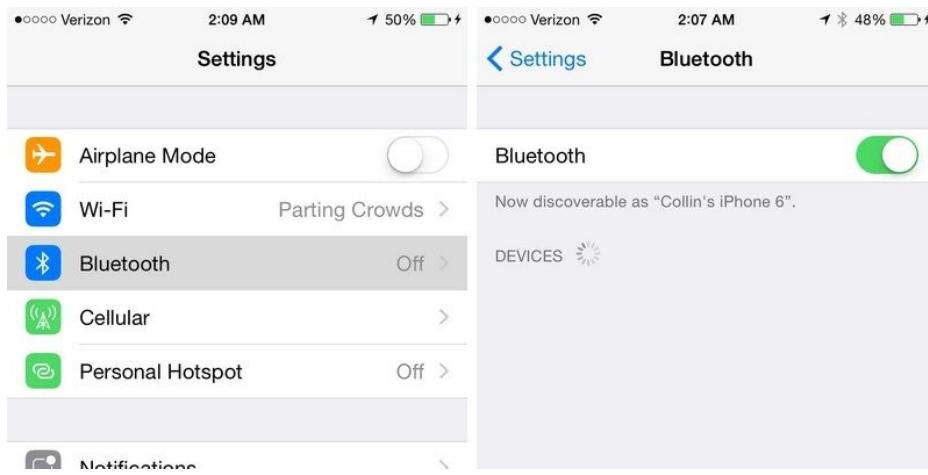
The app is compatible with these BLE devices from Adafruit, and possibly more:

- [Bluefruit LE nRF8001 Breakout](http://adafru.it/1697) (<http://adafru.it/1697>)
- [Bluefruit LE Friend](http://adafru.it/2267) (<http://adafru.it/2267>)
- [Flora Wearable Bluefruit LE Module](http://adafru.it/2487) (<http://adafru.it/2487>)
- [Adafruit Bluefruit LE SPI Friend](http://adafru.it/2633) (<http://adafru.it/2633>)
- [Adafruit Bluefruit LE Micro](http://adafru.it/2661) (<http://adafru.it/2661>)
- [Adafruit Feather 32u4 Bluefruit LE](https://adafru.it/keO) (<https://adafru.it/keO>)
- [Adafruit Feather M0 Bluefruit LE](https://adafru.it/t6a) (<https://adafru.it/t6a>)
- [Adafruit Feather nRF52 Bluefruit LE - nRF52832](https://adafru.it/vAx) (<https://adafru.it/vAx>)
- [Adafruit Feather nRF52840 Express](https://adafru.it/DLQ) (<https://adafru.it/DLQ>)

First off - install the app from one of the App stores listed above if you haven't already.

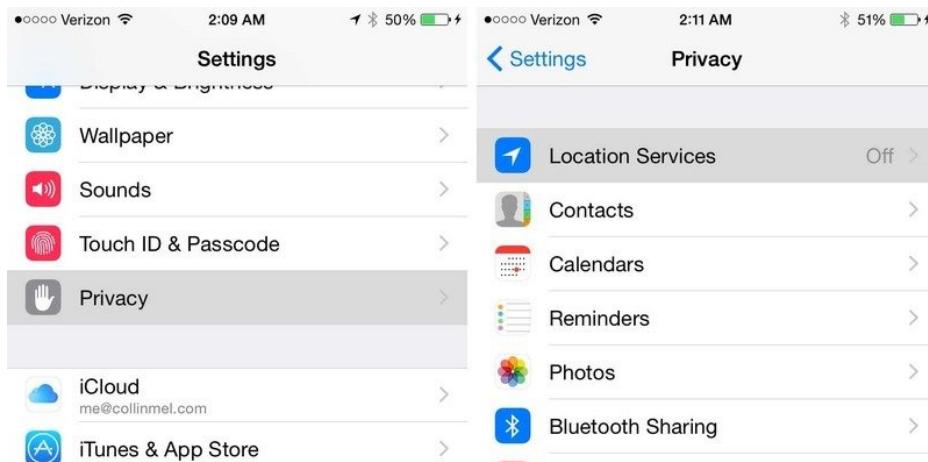
Enable Bluetooth

If Bluetooth is disabled on your device, enable it by going to **Settings->Bluetooth** on your iOS device, or the analogous setting on your Android device.



Enable Location Services

If you plan to use the app to send location/GPS data to Bluefruit LE, enable Location Services. Enable it on iOS using **Settings->Privacy->Location Services**.



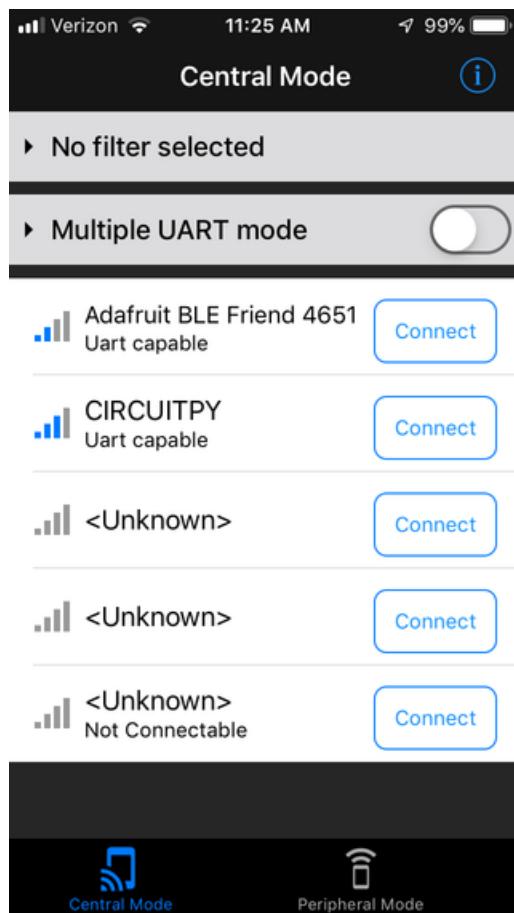
Scan for Devices

On launch, the app will automatically begin to scan for nearby Bluetooth LE devices. Devices are presented in a table view in the order in which they were discovered.

Don't forget to turn on Bluetooth on your device! Airplane Mode turns off BLE.

iPhone

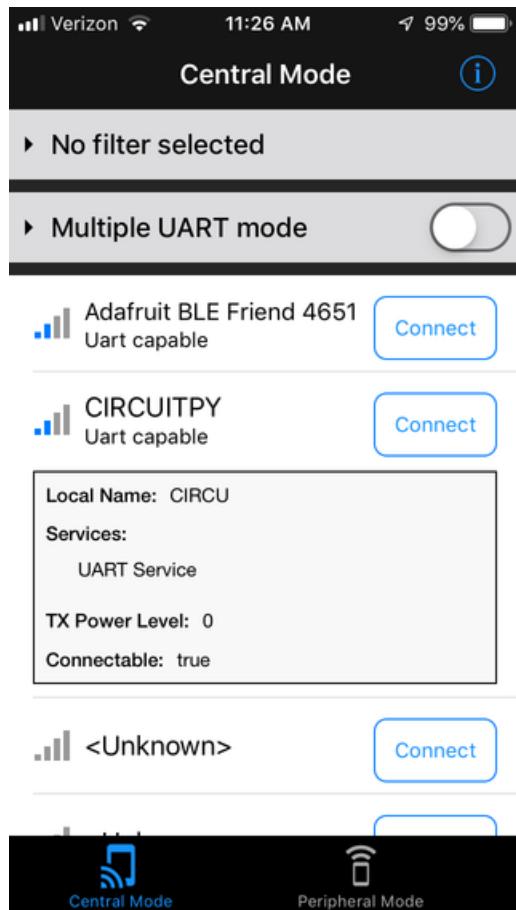
The following images depict the app when used on the iPhone. The Android app looks similar.



The device list will display all BLE devices discovered by the app (not just Bluefruit hardware) - so you may see a quite a few "" entries for devices that don't advertise their name, as seen above.

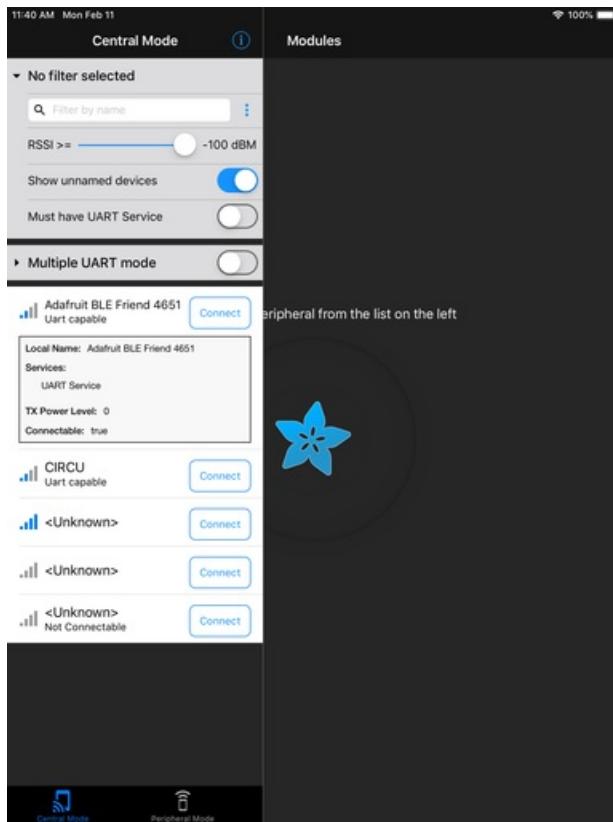
- To refresh the list and start a new scan, simply swipe down on the current list.
- Each device's signal strength is displayed in the left side of its row.

If you tap on the device entry (not on Connect), you'll see more detail about a particular device:



iPad

The following images depict the app when used on the iPad

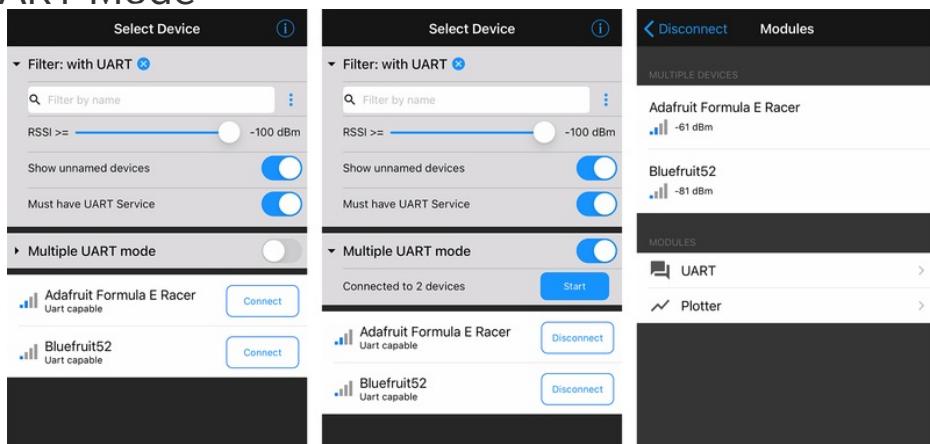


Tap the middle of a device's table row to reveal its relevant advertisement data.

- Any device listed with a "Connect" button at the right can be accessed in Info mode.
- Any device listed as "UART Capable" can be used with all modes - Info, UART, Pin I/O, & Controller.

To use the Bluefruit Connect app with your device it must be "UART capable"! All Adafruit BLE devices implement the UART interface, but other devices may not

Multiple UART Mode



The Multiple UART feature allows to monitor incoming data from, and send data to multiple devices simultaneously.

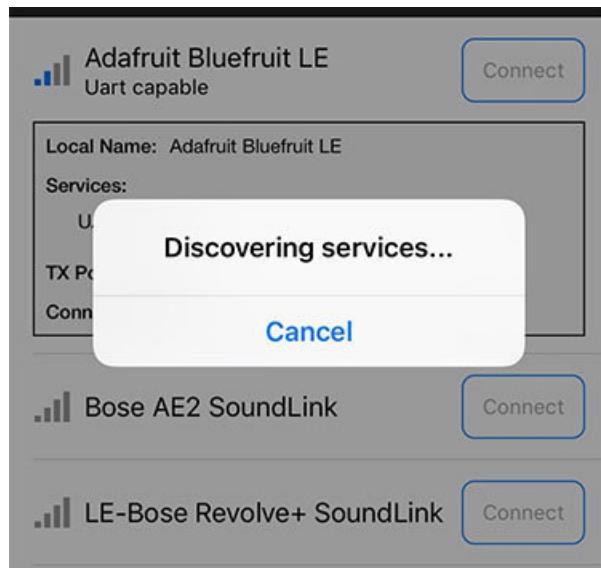
To use it:

- Activate the **Multiple UART Mode** switch
- Tap **Connect** next to each device you'd like to include
- Tap the Start button below the Multiple UART mode to begin.

Once connected, you can choose **UART** or **Plotter** module to view incoming data from all connected peripherals. In the **UART** module, you can send data to one or all connected devices at once.

Connect

Tap the Connect button on the **UART** capable list entry you wish to use and choose a connection mode from the menu that appears.



Troubleshooting



If you're having trouble finding your **Bluefruit** device in the **scanned peripherals** list, ensure the board is **powered** and **not paired** with any other BLE devices. If the problem persists, it could be due to **caching** issues in the iOS or Android operating system. For a fix, try the following:

- **Cycle Bluetooth** - Turn your mobile device's Bluetooth radio off and on again in the Settings app.
- **Relaunch App** - Quit the **Bluetooth LE Connect** app and restart it. (instructions for **iOS** (<https://adafru.it/DMn>) & **Android** (<https://adafru.it/DMo>))

- **Cycle Power** - Restart your mobile device by powering it off and restarting.

Doing one or both of the above solves most peripheral scanning issues. If you're still having trouble, try searching the [Adafruit Support forum \(<https://adafru.it/DMI>\)](https://adafru.it/DMI) for your issue.

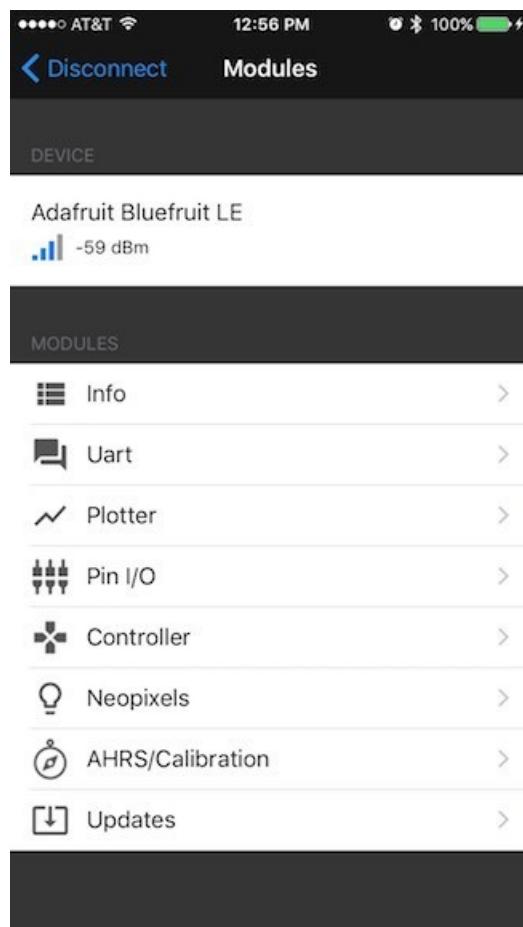
Features

Even though the Bluefruit Connect app only uses **UART** as the *transport* for sending and receiving data to/from your BLE device, it has **multiple interaction modes**

Each of these modes can do different things and let you interact in a unique way.

Don't forget! All of these modes use the **UART Service**, but present the data in a different way. On the hardware side, your firmware will have to know what it is expecting and sometimes may need to parse the data coming back from the app.

For example, if using the **Color Picker**, the app will send the color data in a mini packet. If using the button controller, you'll get button presses/releases in packets instead.



Available Modes:

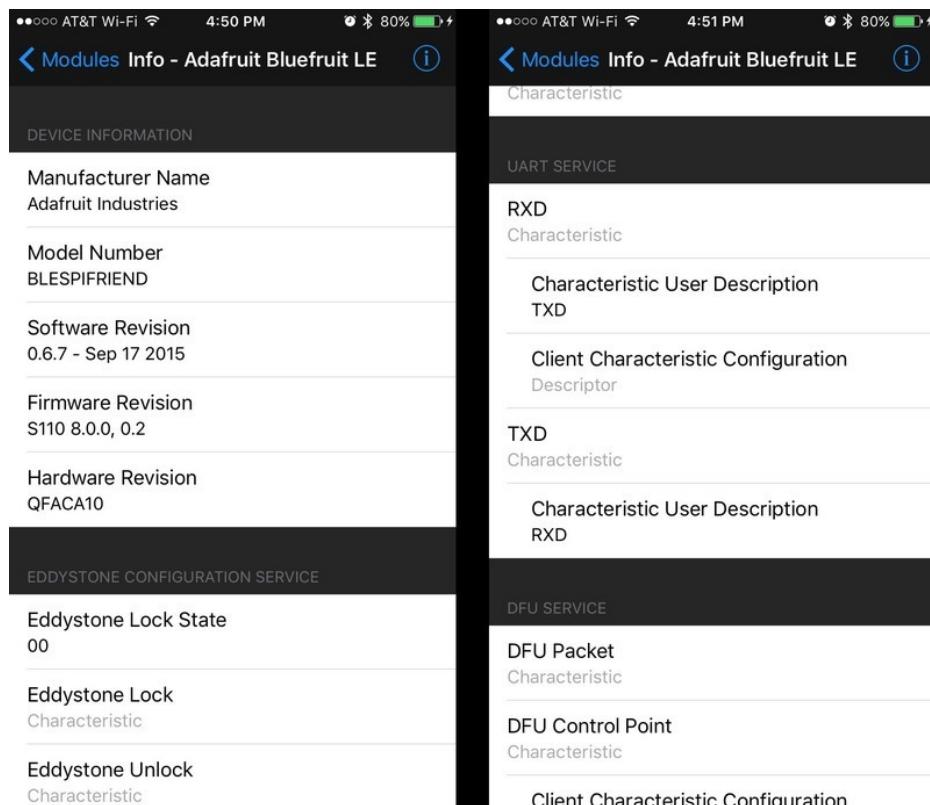
- [Info](https://adafru.it/BYg)
- [UART Terminal Mode](https://adafru.it/xSF)
- [MQTT](https://adafru.it/yb7)
- [Plotter](https://adafru.it/xTa)
- [Controller](https://adafru.it/iCI) ([Data Streaming](https://adafru.it/DN9), [Control Pad](https://adafru.it/DN9), and [Color Picker](https://adafru.it/DN9))
- [AHRS/Calibration](https://adafru.it/Gge)
- [Updates](https://adafru.it/BYh)

- [Image Transfer \(https://adafru.it/Gjc\)](https://adafru.it/Gjc)

Info

Connecting to a peripheral in Info mode will display its Generic Attribute Profile (**GATT**) in the form of a table.

- This mode is available for all connectable BLE devices and can be helpful for learning, troubleshooting, and general curious snooping.
- Tapping on a service row will reveal that service's included characteristics.



To learn more about Bluetooth Services & Characteristics, be sure to check out the [Introduction to Bluetooth Low Energy guide](https://adafru.it/iCp) (<https://adafru.it/iCp>).

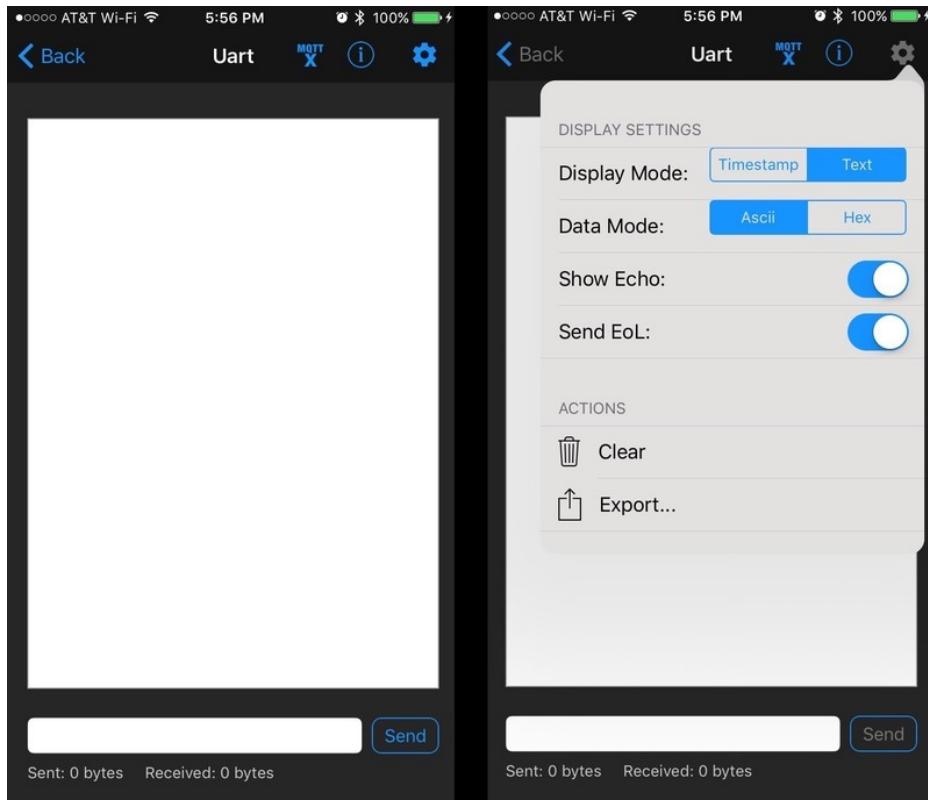
UART Terminal

The UART Terminal mode provides a classic 'serial terminal' interface for sending and receiving strings from a Bluefruit LE device.

It's perfect for sending and receiving data without any interpretation.

Note: This mode can be used in conjunction with [Bluefruit LE Friend's Command mode](#) (<https://adafru.it/iCO>) to configure or get additional info about the device.

Main Window



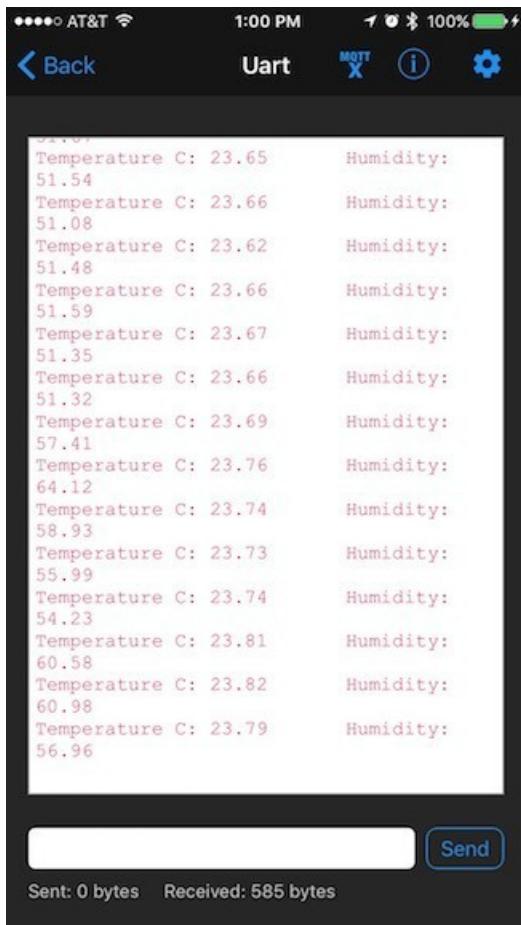
The main log window is in the middle, and will display data both received and, if Echo is on, sent.

Echo

Toggle the **Echo** switch in the settings the upper right to also display outgoing data sent from the app.

Data received from the remote BLE device appears in red.

Data sent from the app appears in blue.

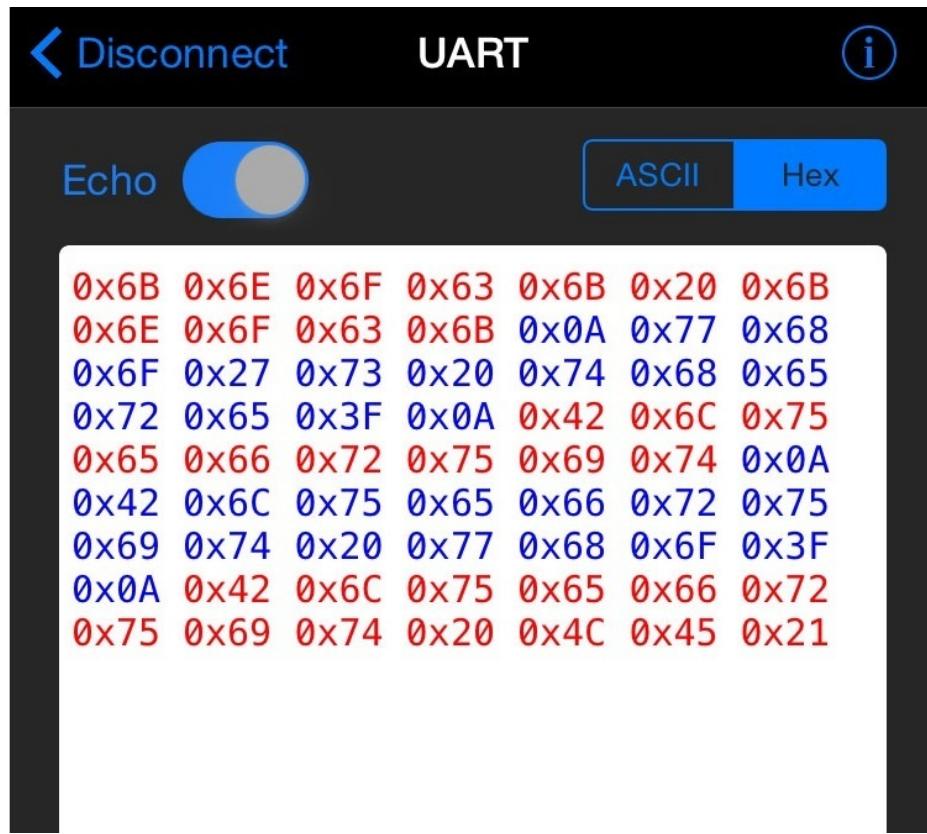


ASCII / HEX

The display format of the log window can be controlled using the **ASCII/Hex** switch in settings in the upper right.

ASCII will do its best to translate the data to 8-bit human-viewable text characters.

Hex will give you Oxnn formatted bytes, still color coded.



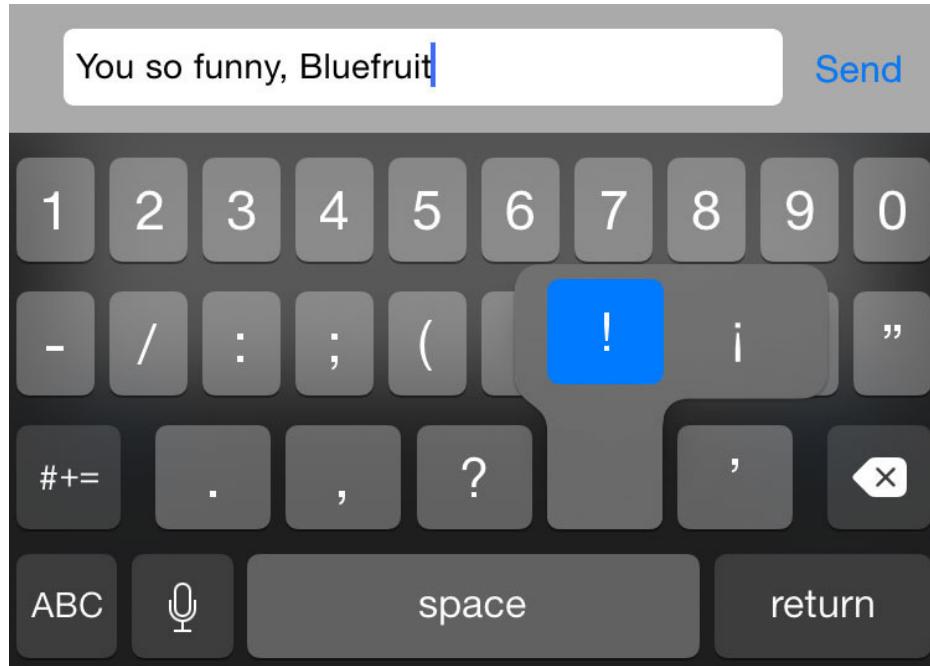
Copy & Clear

Tapping **Copy** will copy all text from the log window to the iOS clipboard.

Tapping **Clear** will delete all text from the window.

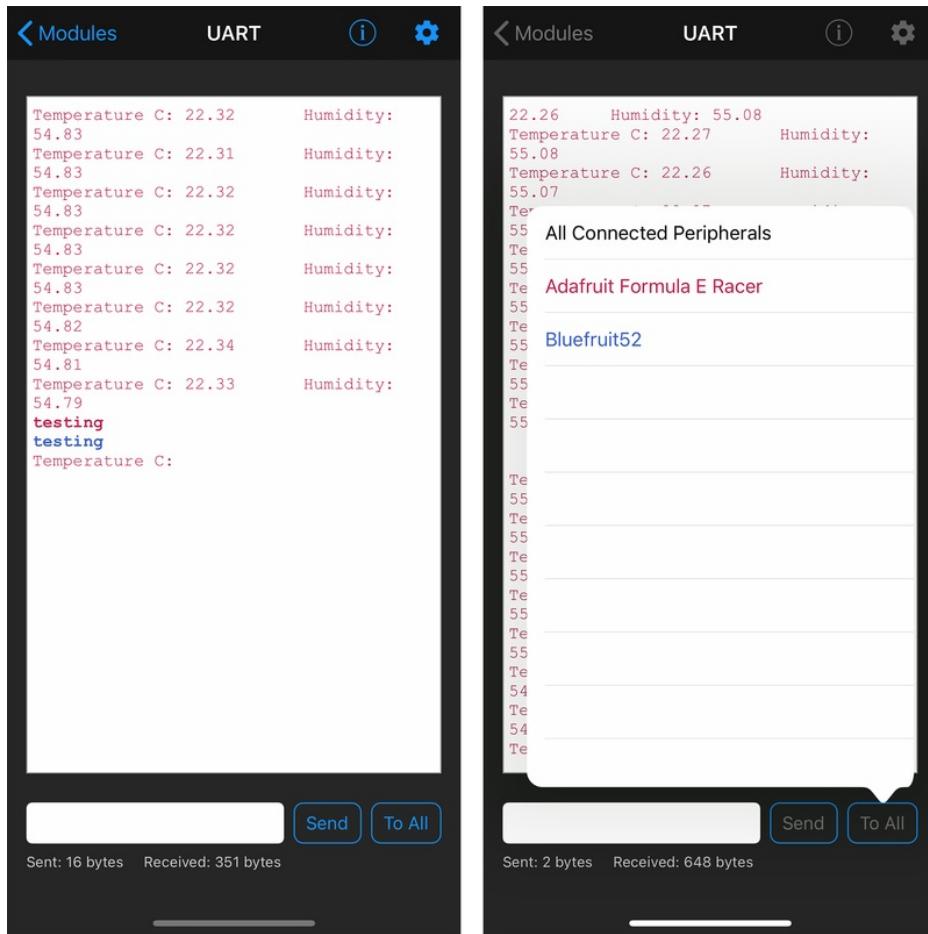


Sending Strings



- Tap the text field at the bottom of the screen to bring up the keyboard and begin composing a new string to send.
- Newline characters can be added using the return key.
- Press the **Send** button to send the string over UART to your Bluefruit LE device.
- To hide the keyboard, simply tap the log window while the keyboard is shown.

Multiple UART Mode



When connected to more than one Bluefruit LE device in **Multiple UART** mode, you'll see an **All** button to the right of the **Send** button. Tap the **All** button to choose which specific devices you wish to send your inputted string to.

MQTT



MQTT stands for Message Queue Telemetry Transport. It's a protocol designed for low-bandwidth, high latency networks. You can learn more about it [here](https://adafru.it/yqD). (<https://adafru.it/yqD>)

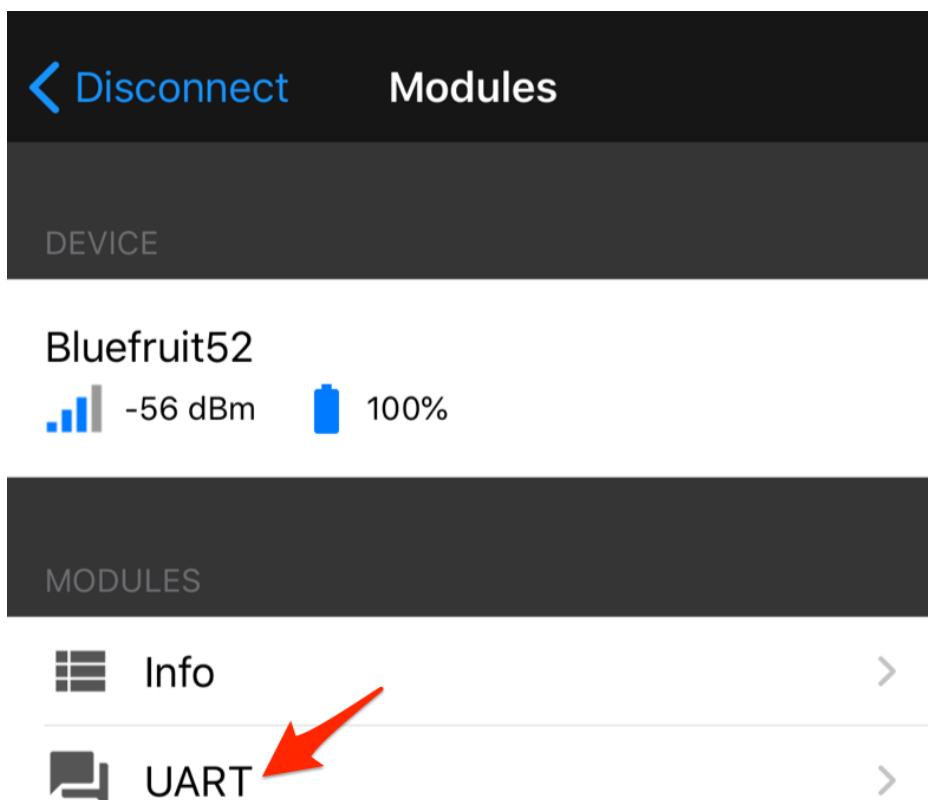
The Bluefruit LE Connect App allows you to send and receive data using the MQTT protocol. This may be useful for making sensor readings from a Bluefruit device viewable on the web.

Configuration

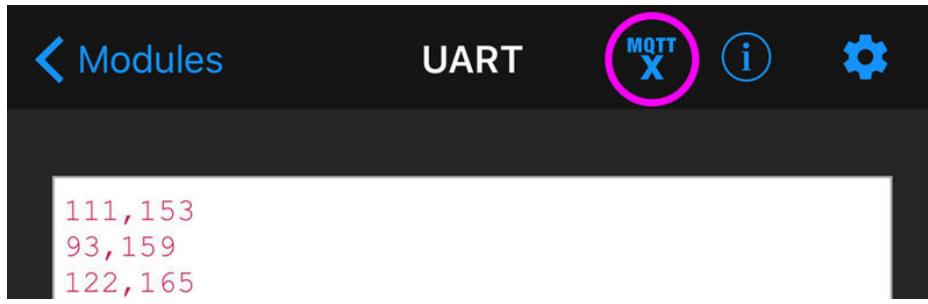
You will need an [adafruit.io](https://adafru.it/fJs) (<https://adafru.it/fJs>) account in order to log and collect your data. You can make one [here](https://adafru.it/eZ8) (<https://adafru.it/eZ8>).

- To learn more about adafruit.io, [check out the guide for it here](https://adafru.it/JWd) (<https://adafru.it/JWd>).

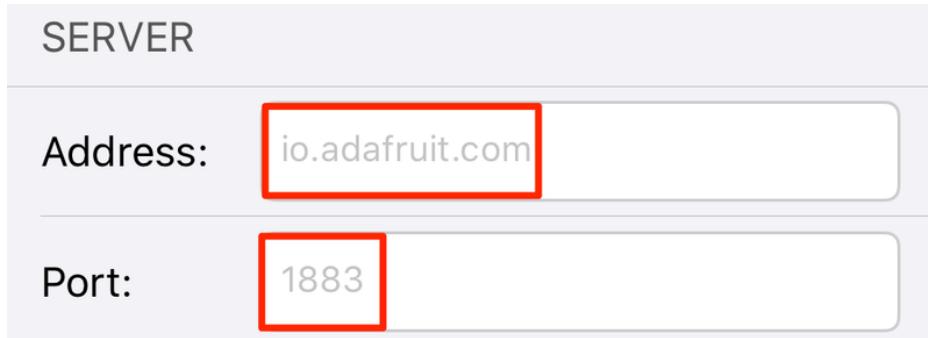
Once you have your Adafruit IO account set up, connect to your device using the Bluefruit LE Connect app. Tap the **UART** mode.



In the UART module, **tap the MQTT button** at the top of the screen.

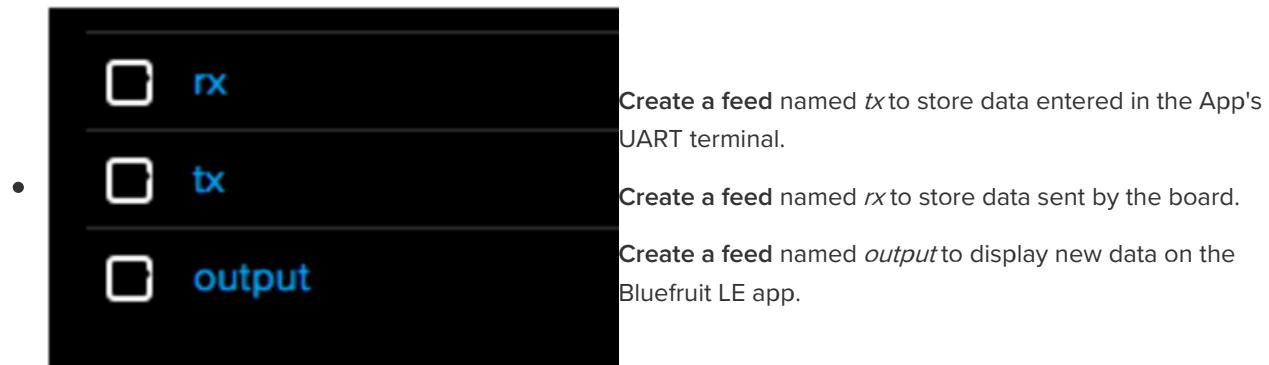


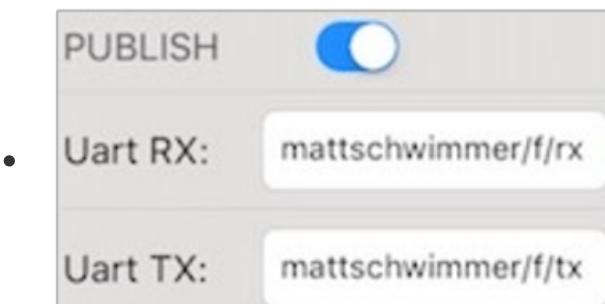
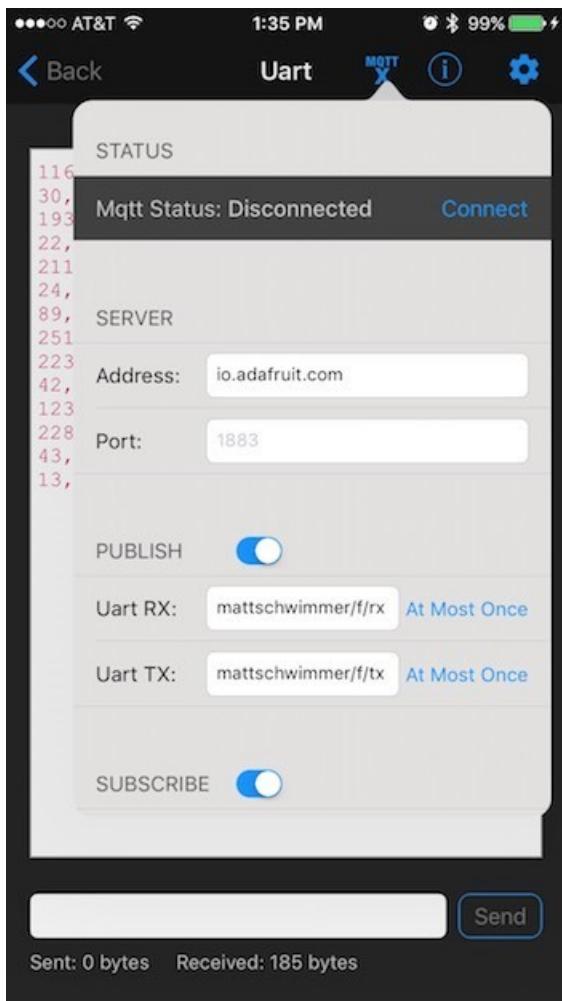
Change the Address to io.adafruit.com. Set the Port to 1883.



Before filling in the necessary information, you will need to create 3 feeds on your adafruit.io account: *rx*, *tx*, and *output*.

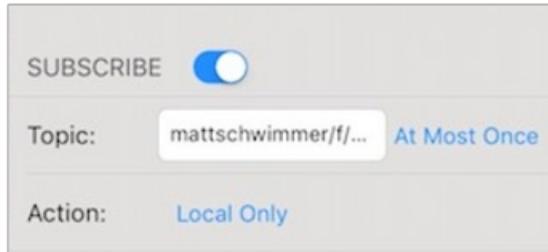
- If you do not already know how to create a feed, head over to [Adafruit IO Basics: Feeds](#) (<https://adafru.it/ioA>).





Under PUBLISH,

- Set Uart RX to your_username/f/rx
- Set Uart TX to your_username/f/tx

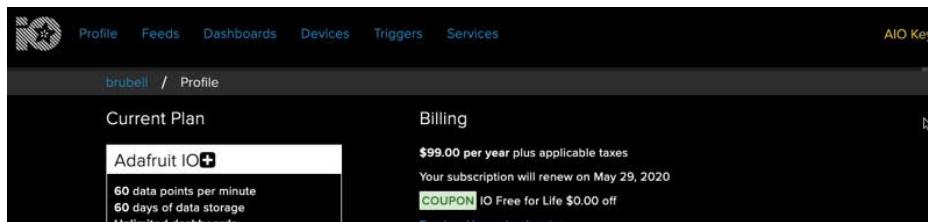


Under SUBSCRIBE,

- Set Topic to
your_username/f/output

You are also going to need your Adafruit IO username and secret API key.

Navigate to your profile (<https://adafru.it/EpC>) and click the View AIO Key button to retrieve them. Write them down in a safe place, you'll need them for the next step.



Under ADVANCED, enter your Adafruit IO username and key.

Do NOT enter your Adafruit IO password! Use your Adafruit IO KEY

ADVANCED

Username: brubell

Pass/Key: aid[REDACTED]

Usage

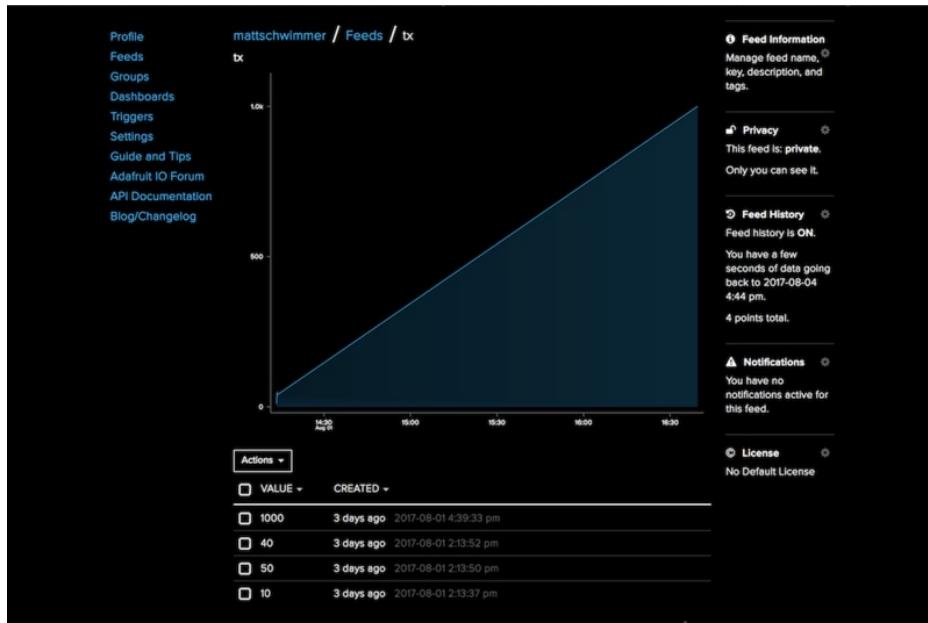
Here are some examples of feeds created using the Bluefruit LE Connect app in the same manner as described above.

Received Data



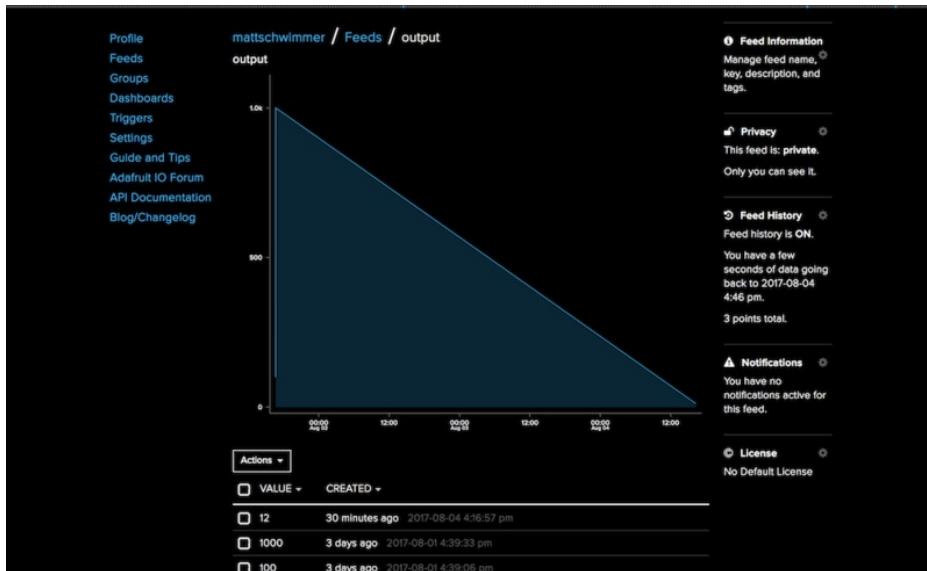
This is an example of an RX feed that presents data **sent** from an Adafruit Feather Bluefruit and is **received** by iPhone using the Bluefruit LE Connect app via MQTT. The data is sent using an ASCII Numeric format similar to how plotter data is sent. You can see the plotter feature [here](https://adafru.it/xTa) (<https://adafru.it/xTa>).

Written Data



This is an example of a TX feed that presents data **sent** from the Bluefruit LE Connect app to adafruit.io via MQTT.

Output Data



This is an example of an output feed that presents data created on adafruit.io [sent](#) to the Bluefruit LE Connect app via MQTT

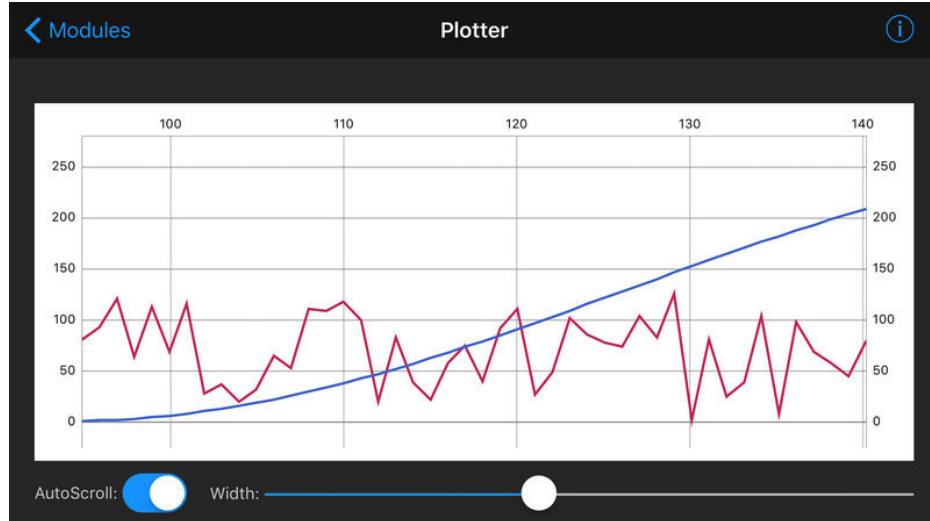
Plotter

The Plotter mode allows for users to visualize data received from Bluetooth LE compatible devices.

This data is sent **from the Bluefruit device** to the phone/tablet.

Feel free to test the plotting feature using [this demo sketch \(https://adafru.it/xTc\)](https://adafru.it/xTc)!

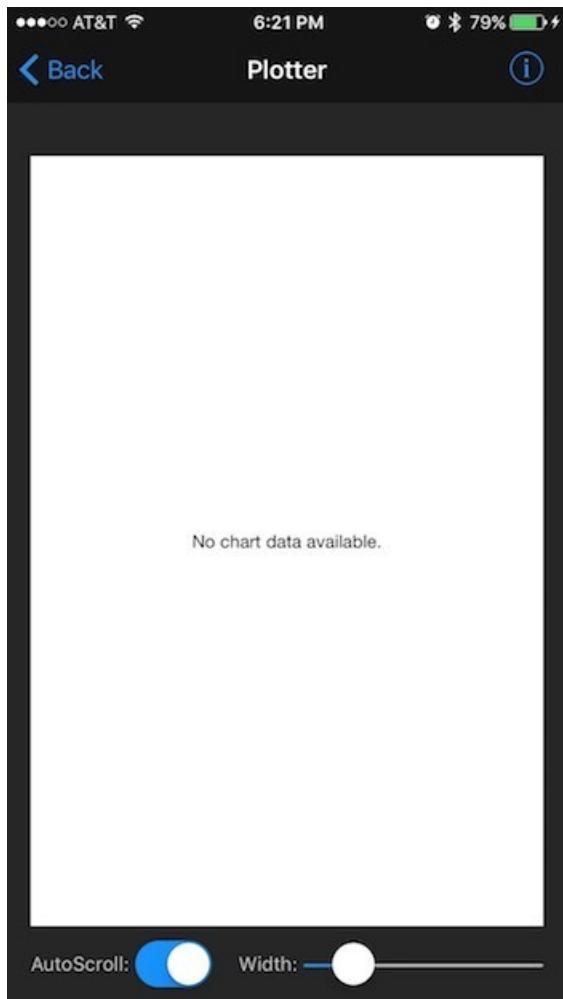
If you're running into issues sending data to the Plotter, try limiting throughput to 10 values per second.



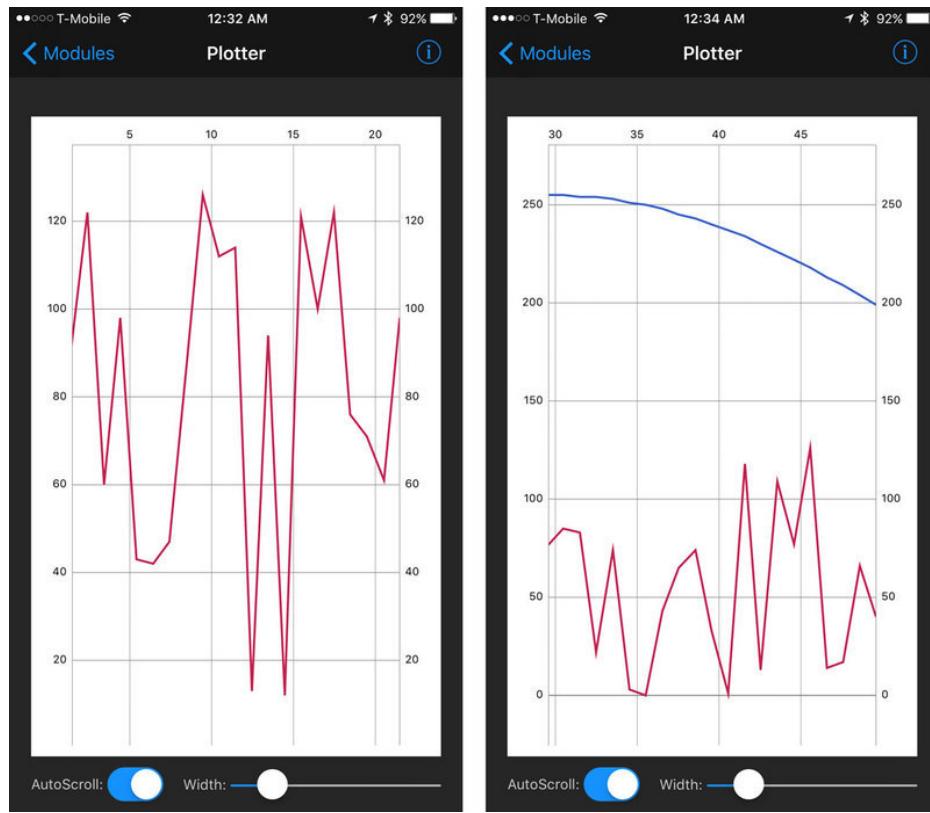
Main Plotter View

The main plotter window does not display X and Y axes unless the Bluefruit Connect app receives the necessary data to plot.

This is what the Bluefruit LE Connect app displays by default if there is no data to plot.



Once the required data is received, the plotting begins!

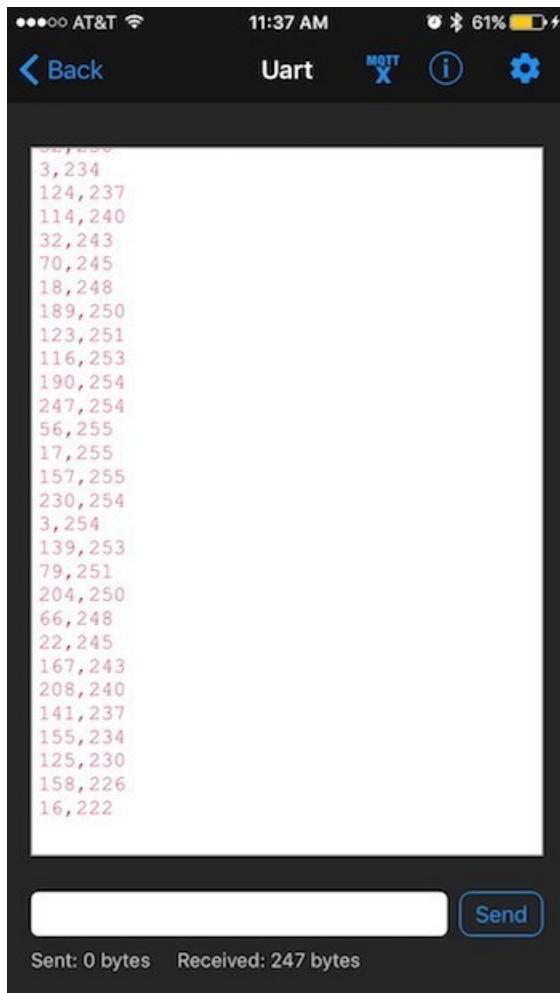


If your project utilizes multiple data streams, Bluefruit LE Connect will plot both for you! Above, notice that the graph on the right shows a second set of data.

Formatting

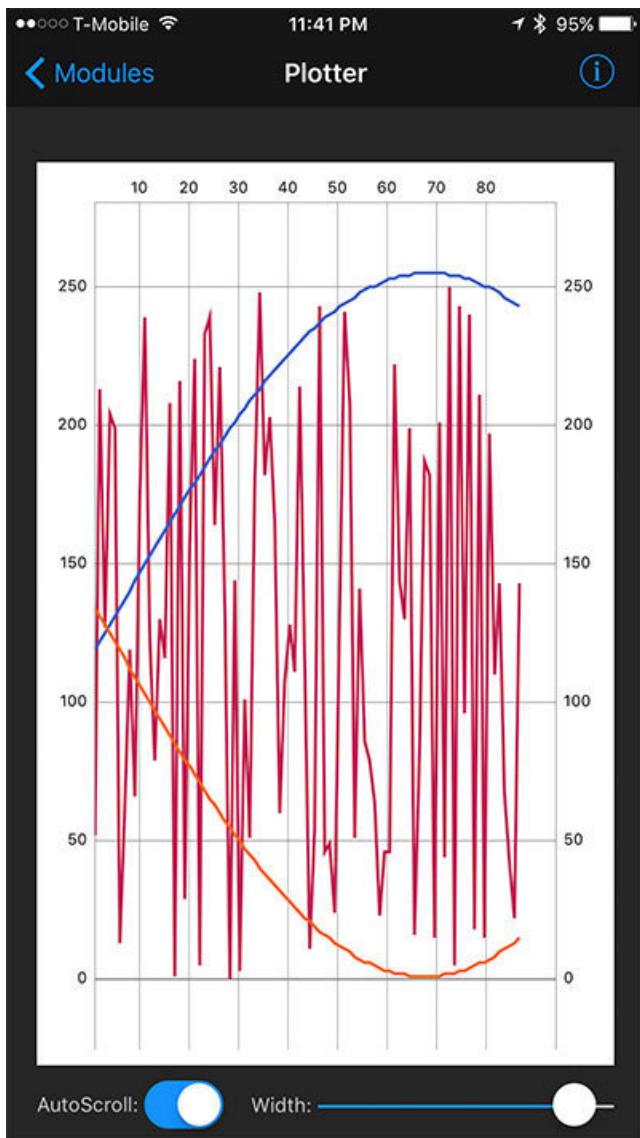
The app interprets incoming numeric values in ASCII format. Separate data values should be followed by a comma or tab character. Separate each set of values sent using a newline character or simply use `ble.println()` (which automatically appends a newline char). This allows Bluefruit LE Connect to know when to plot the next set of values.

For example, a properly formatted stream of plotter values will look like this when viewed in the app's [UART terminal \(<https://adafru.it/xTd>\)](https://adafru.it/xTd):



If you need to plot more data streams, simply add an additional comma followed by another numeric value.

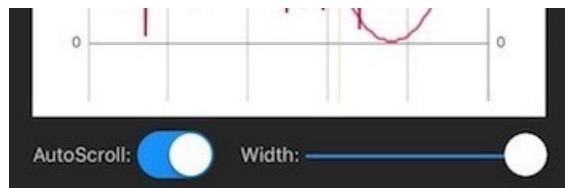
A plot utilizing 3 data streams would look something like the picture shown below:



The plotter also supports landscape view!



AutoScroll and Plot Width



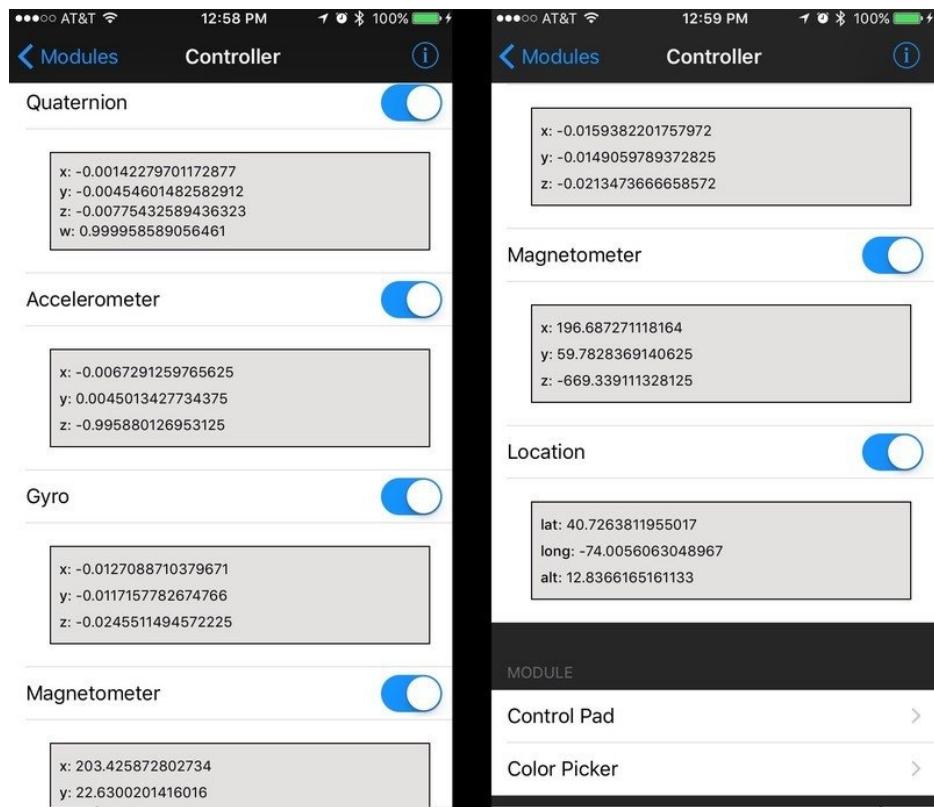
When switched on, AutoScroll will adjust the graph size and follow the most recent data collected by the Bluefruit LE Connect app

When turned off, swipe left and right to scroll through your data. Use the Width slider to adjust the width of the graph.

Controller

The Controller mode provides a variety of ways to control your Bluefruit LE enabled project including Sensor Data, Control Pad, & Color Picker.

This data is sent **from the phone/tablet** to the Bluefruit device.



Format for Sent Data

For an example of how to parse the data on Arduino, check out the [BLE_Controller_Test code on github](#) (<https://adafru.it/exz>).

- Each Controller data packet sent is **prefixed** with single byte char “!” (0x21) followed by a single byte char initial for identification.
- Sensor data values are encoded as **floats of 4 byte length**.
- Each packet ends with a single byte checksum for validation.

Checksum

The single-byte checksum that appends each Controller data packet is calculated by adding all previous bytes of the packet and then inverting the sum.

An example of how to use the checksum to validate a Controller packet can be found in the [BLE_Controller_Test](#) (<https://adafru.it/exz>) Arduino sketch:

```
boolean checkCRC(uint8_t *buffer) {
    uint8_t len = sizeof(buffer);
    uint8_t crc = buffer[len-2];
    uint8_t sum = 0;
```

```
for (int i = 0; i < (len-1); i++) {
    sum += buffer[i];
}

Serial.print("CRC ");

if ((crc & ~sum) == 0) {
    Serial.println("PASS");
    return true;
}

else {
    Serial.println("FAIL");
    return false;
}

}
```

Sensors

The top section of the Controller table lists the available types of sensor data which can be streamed from your iOS device. Tap the button at the right of each sensor row to begin streaming its relevant data.

- All sensor data updates, except for Location, are sent out over BLE ten times per second.
- Location updates are sent whenever GPS data changes, or every 30 seconds if no change occurs.
- **Quaternion** - sends iOS Device Motion data to describe device attitude. This data is derived from Accelerometer, Gyro, and Magnetometer readings.

Prefix: **!Q**

Format:

```
[!] ['Q'] [float x] [float y] [float z] [float w] [CRC]
```

- **Accelerometer** - sends raw accelerometer data.

Prefix: **!A**

Format:

```
[!] ['A'] [float x] [float y] [float z] [CRC]
```

- **Gyro** - sends raw gyroscope data.

Prefix: **!G**

Format:

```
[!] ['G'] [float x] [float y] [float z] [CRC]
```

- **Magnetometer** - sends raw, uncalibrated magnetometer data.

Prefix: !M

Format:

[!] [M] [float x] [float y] [float z] [CRC]

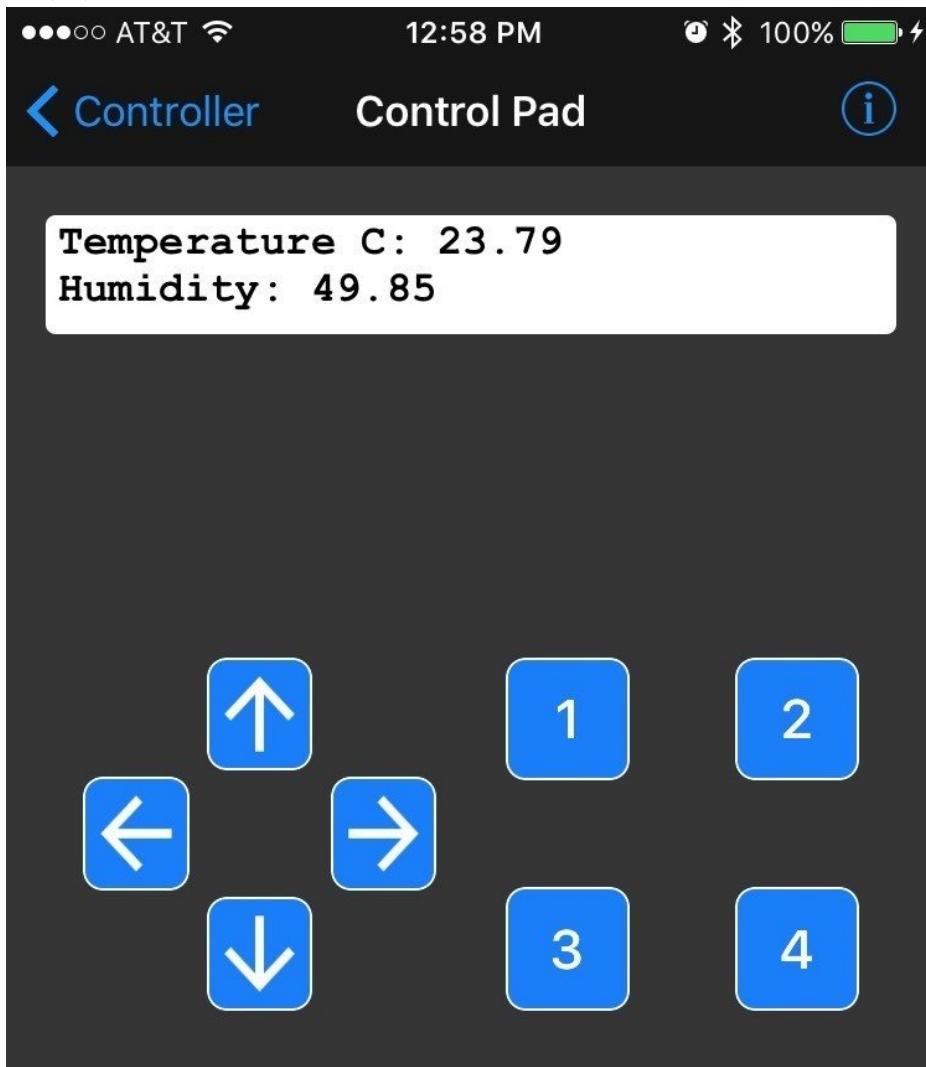
- **Location** - sends GPS data, requires user permission before initial use.

Prefix: !L

Format:

[!] [L] [float lat.] [float long.] [float alt.] [CRC]

Control Pad



The Control Pad function provides a familiar momentary button interface for common control scenarios. Data is sent on the press and release of each button. Each packet consists of 4 bytes, each representing a char value. The first two chars identify the packet as a button message, the third specifies a button, and

the fourth signifies either a press or release.

Prefix: !B

Examples:

Button 4 pressed: ['!'] ['B'] ['4'] ['1'] [CRC]

Button 4 released: ['!'] ['B'] ['4'] ['0'] [CRC]

Button Up pressed: ['!'] ['B'] ['5'] ['1'] [CRC]

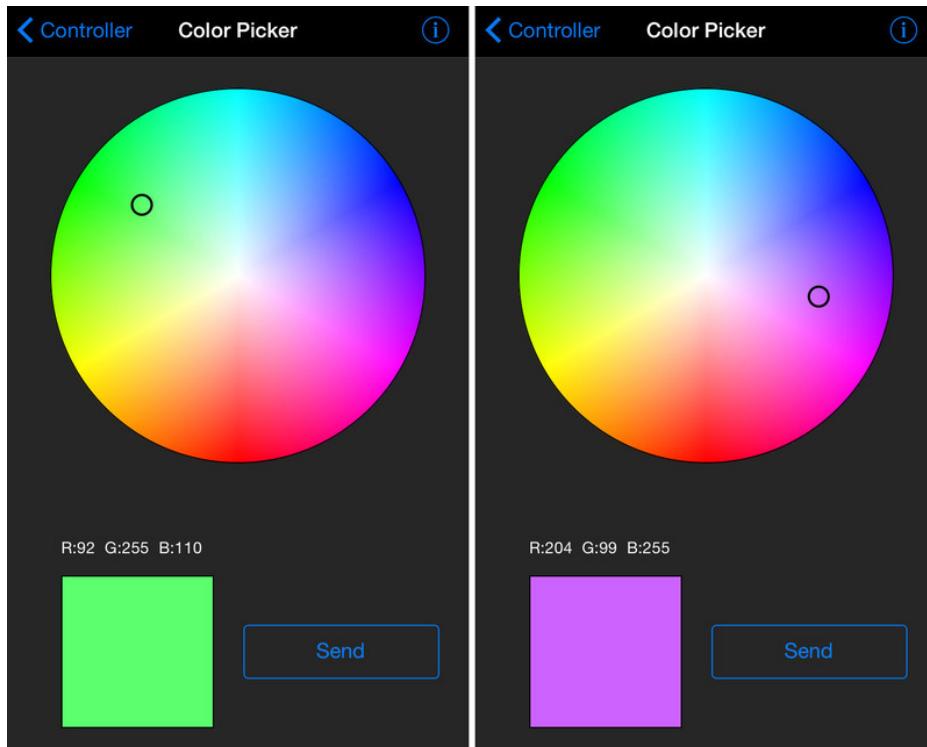
Button Down pressed: ['!'] ['B'] ['6'] ['1'] [CRC]

Button Left pressed: ['!'] ['B'] ['7'] ['1'] [CRC]

Button Right pressed: ['!'] ['B'] ['8'] ['1'] [CRC]

Note: Any activated sensor data streams will continue while using the Control Pad.

Color Picker



The Color Picker sends a color's RGB values to Bluefruit LE. This can be used to control the state of RGB LEDs such as [Neopixels](https://adafru.it/exA) (<https://adafru.it/exA>).

- Touch the color wheel to choose desired color
- Press Send to send the chosen color's red, green, and blue values to Bluefruit via UART in the following format:

Prefix: !C

Format:

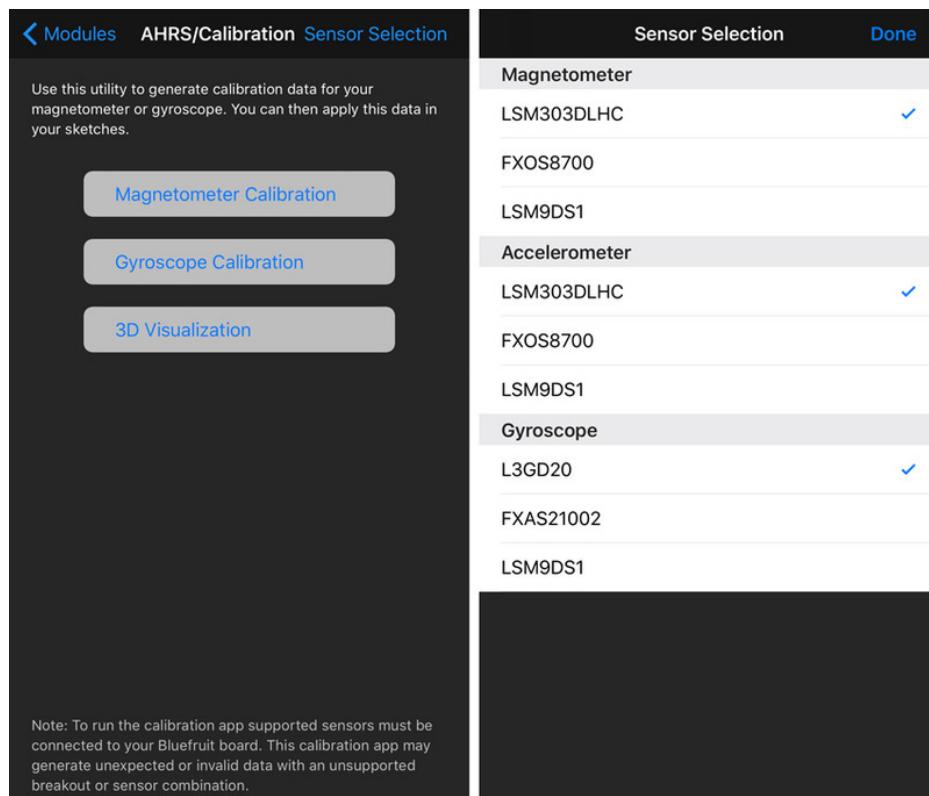
[!] [C] [byte red] [byte green] [byte blue] [CRC]

Note: Any activated sensor data streams will continue while using the Color Picker.

AHRS/Calibration

This feature is still in development

The AHRS/Calibration feature allows you to generate and test calibration code for Magnetometer and Gyroscope sensors connected to your NRF51 based Bluefruit LE device.



Process

Download and run this sketch on your NRF51 powered Bluefruit board:

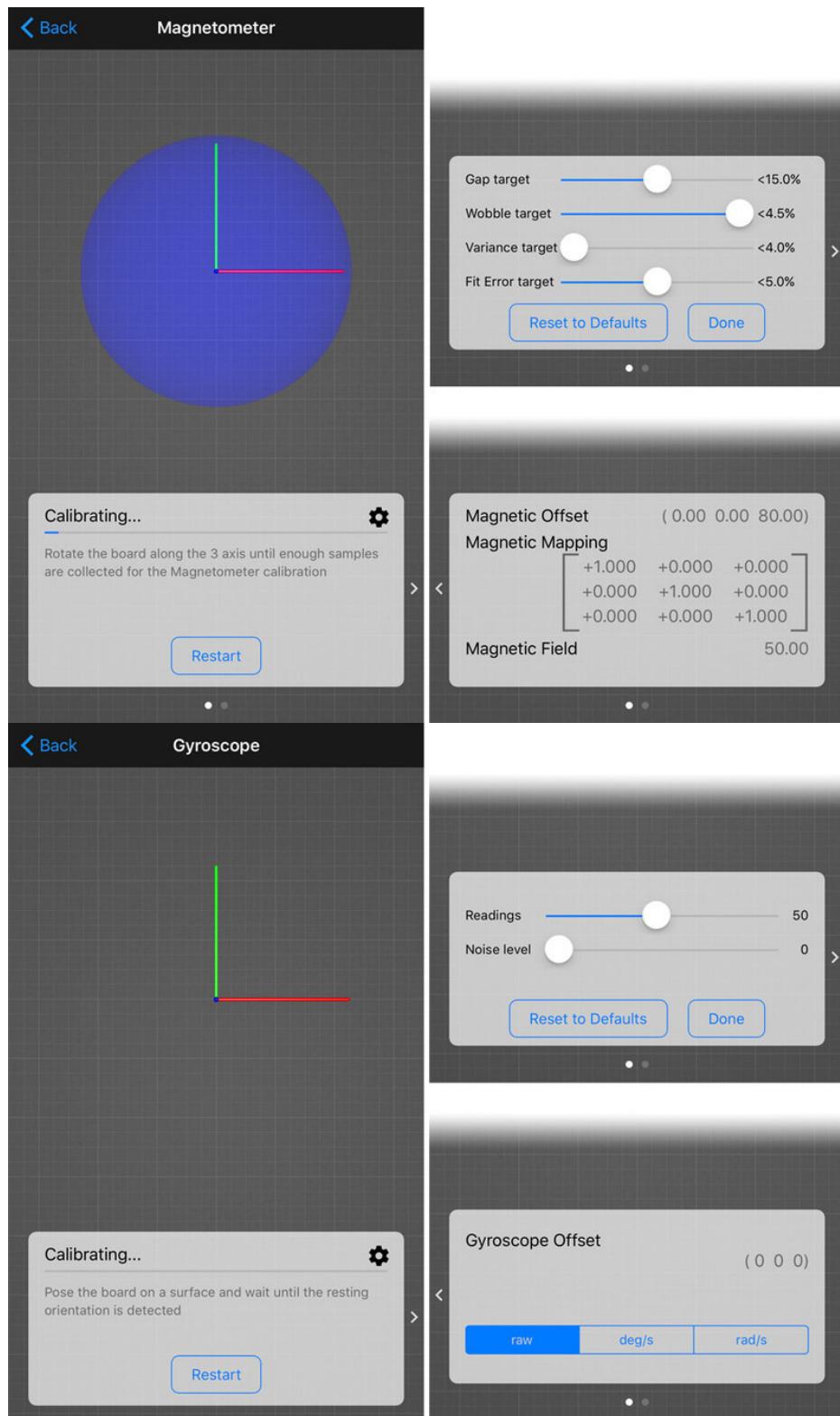
<https://adafru.it/A7T>

<https://adafru.it/A7T>

Connect to your Bluefruit board using the calibration app and choose either **Magnetometer** or **Gyroscope** mode depending on what sensor you'll be calibrating.

Keep rotating slowly in a sphere until you get decent coefficients (this can be a bit tricky and takes some trial and error to get right).

Once you have data, note it down as shown in the USB example above.



Next download this sketch, and plug the mag co-efficient values into it:

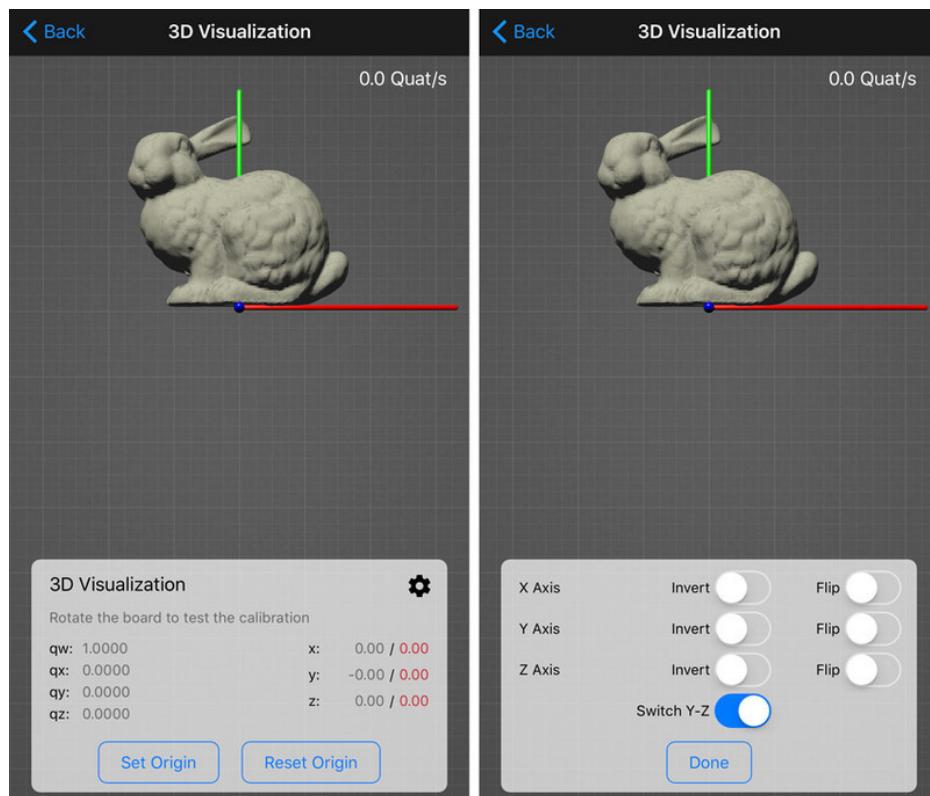
<https://adafru.it/A7U>

<https://adafru.it/A7U>

Once the values are plugged in, upload the sketch to your Bluefruit board.

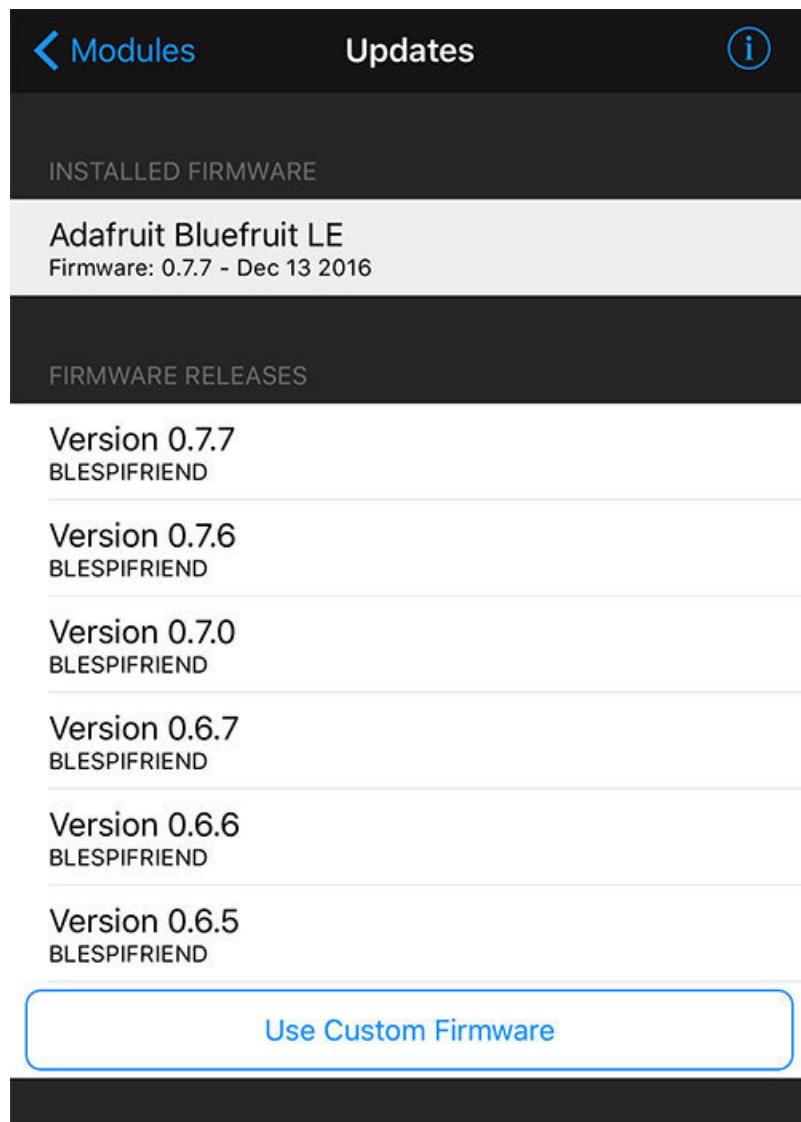
Testing

Connect to your Bluefruit board and run the '3D Visualisation' mode on the app. You will see a 3D model of a bunny. Readings sent from your Bluefruit board should allow you to rotate the model by rotating your board.



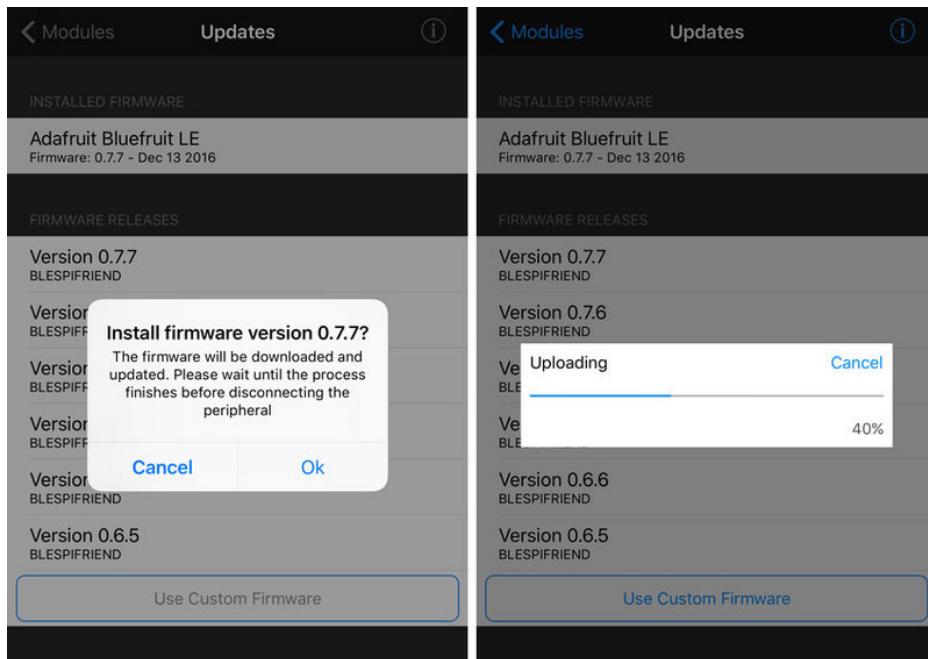
Updates

The Updates features allows you to update the device firmware of your Bluefruit LE hardware over the air. This makes it easy to keep your device firmware up to date.



Updating Firmware

You'll first be presented with a list of compatible firmware versions. Choose the one you want to use (you'll usually want the newest one at the top of the list).



You'll then be asked to confirm the update process. After tapping **Ok**, the process will begin. Once the update process is completed, the app will automatically disconnect from the device and the device will reboot.

Custom Firmware

You can also update your device with an your own unlisted firmware by tapping the **Use Custom Firmware** button. You'll be asked to locate the relevant file (**iOS** users can access files via **iCloud**) and then update process will begin as above.

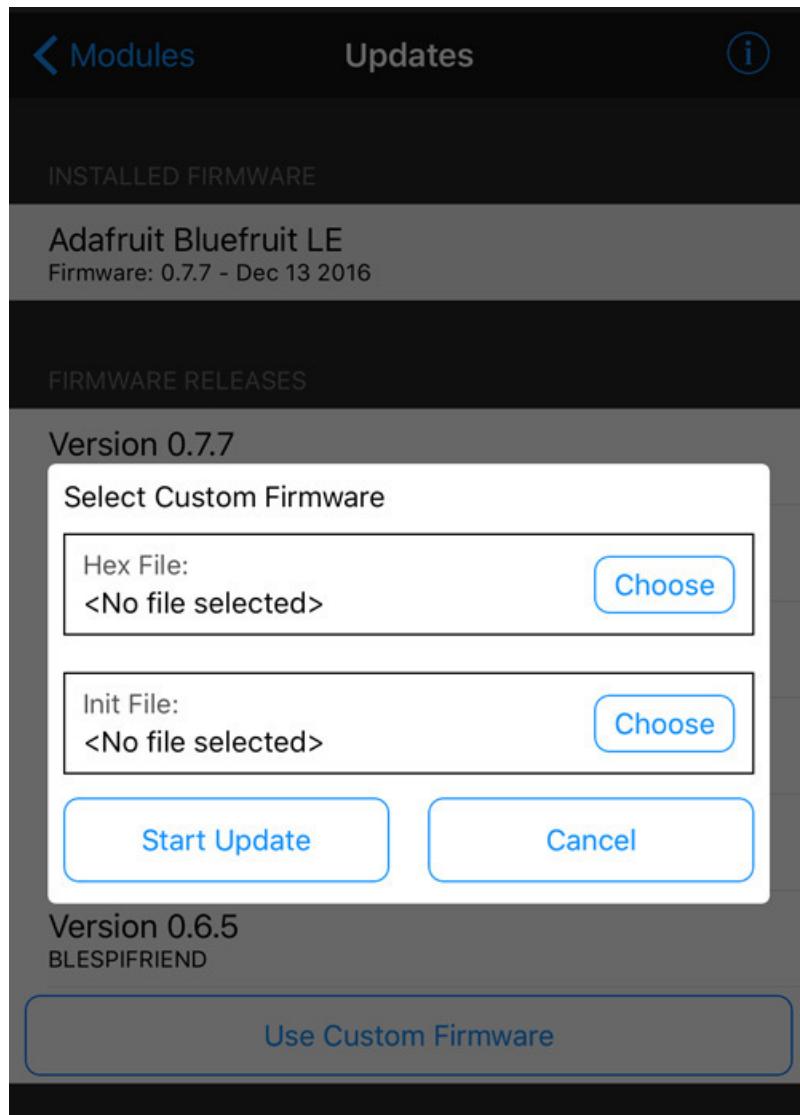
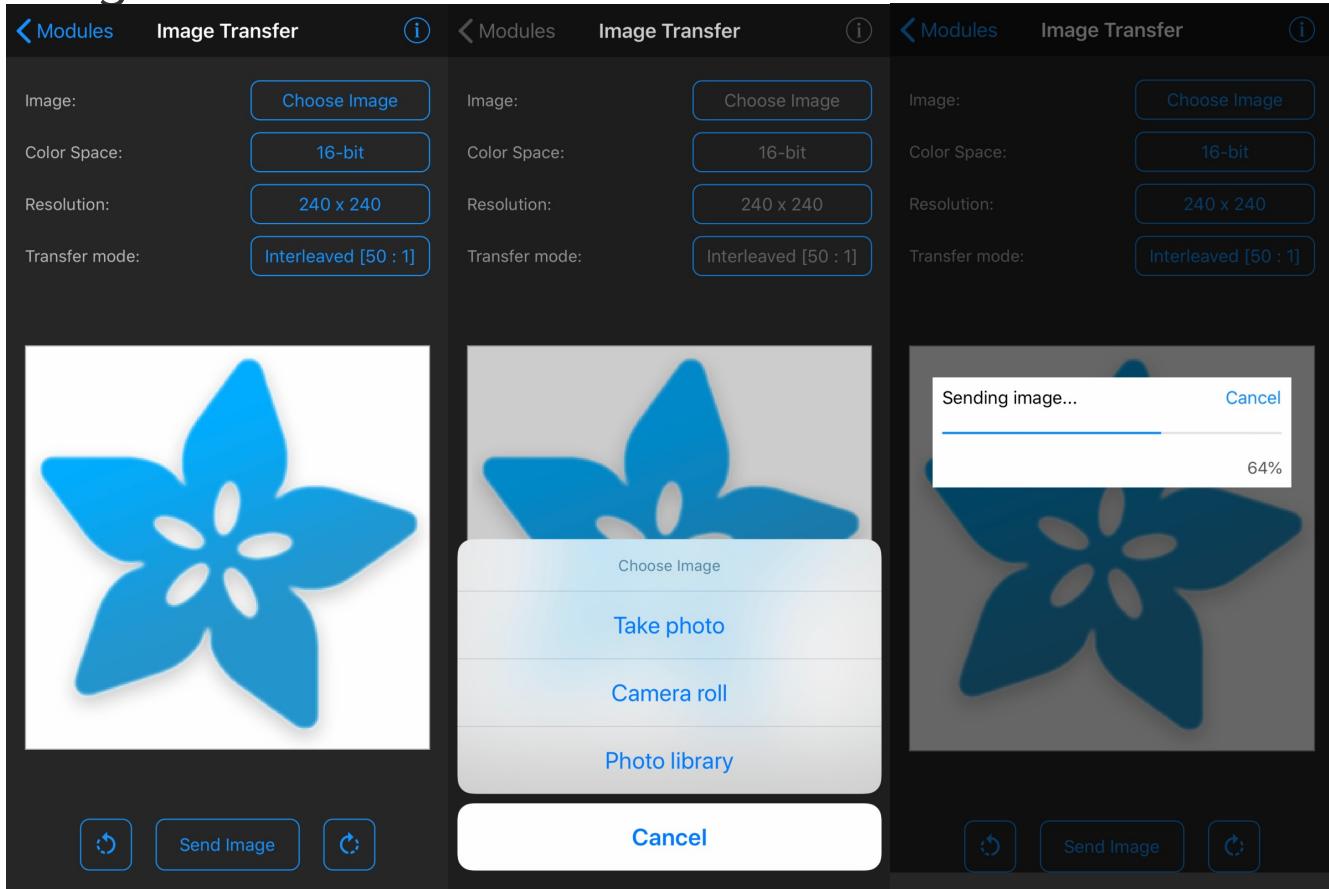


Image Transfer



The Image Transfer mode allows you to select an image and transmit it over BLE. The interface also offers a number of additional features for advanced users.

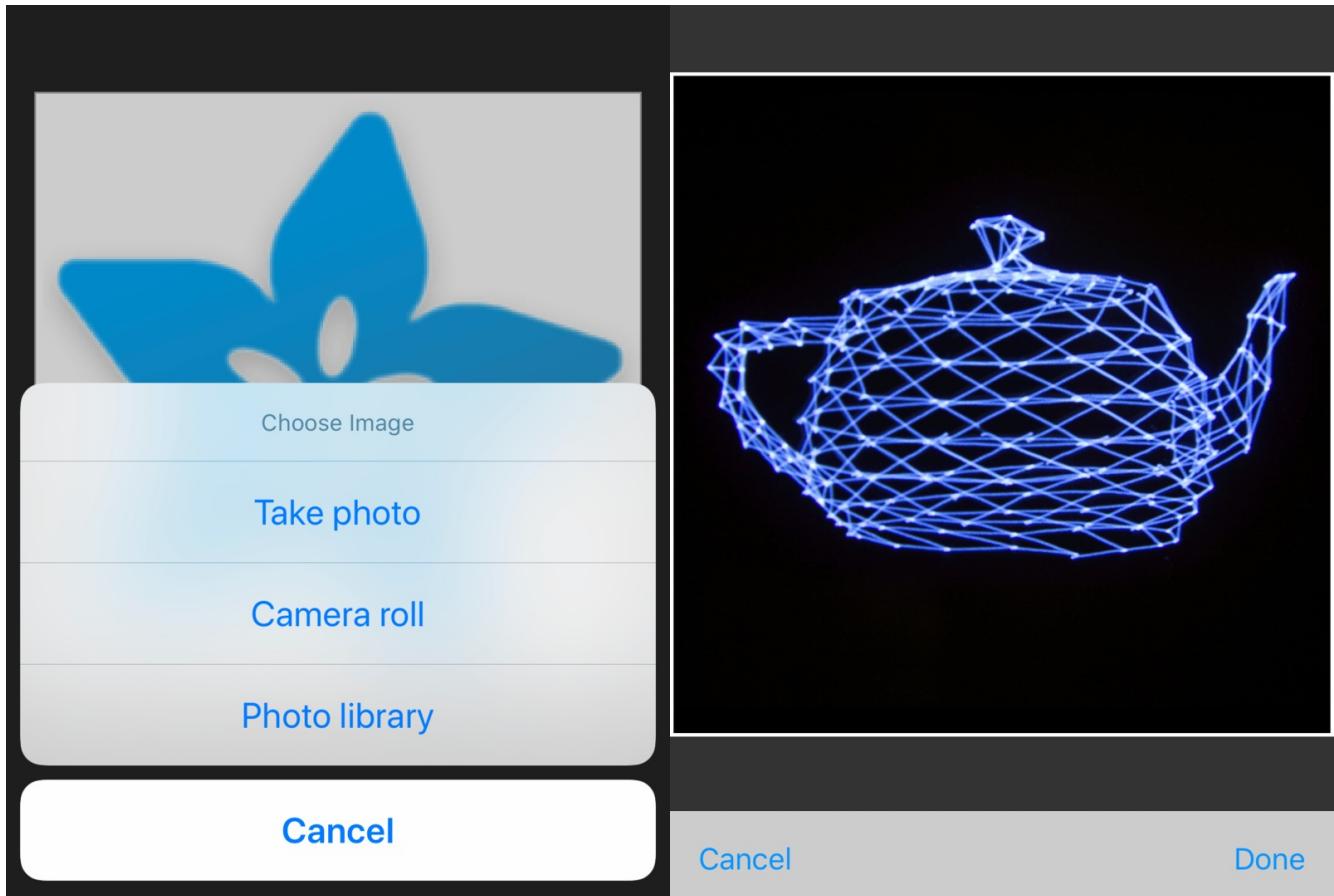
Data Format

Image data is sent from over BLE in the following format:

- Single byte char “!” (0x21)
- Single byte char ‘I’ (0x49)
- Color space (uint8 16 for 16bits 565 or 24 for 24-bit 888)
- Image width (uint16 little endian, 2 bytes)
- Height (uint16 little endian, 2 bytes)
- Pixel data encoded as "RGB 16bit 565" or "RGB 24-bit 888" depending on the Color Space selected
- Single byte CRC

Choose Image





Tap the **Choose Image** button to select an image from your device's photo library or capture a new one from the camera. After selecting your image, use a **pinch gesture** to **scale** it for cropping within the white bounding box.

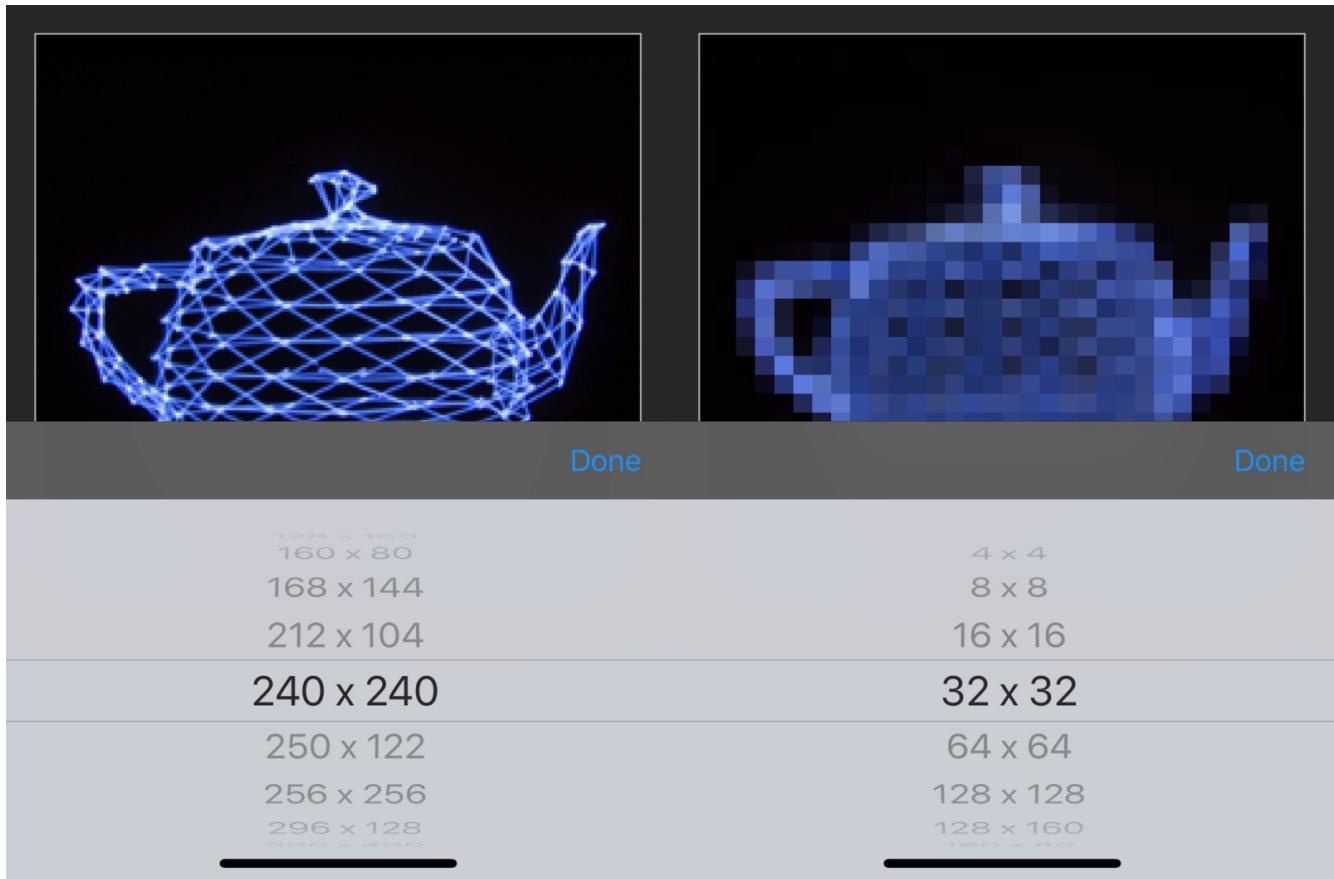
Bit Depth

Bit Depth: **16-bit**

This setting toggles between two color **bit-depth** settings: **16-bit** & **24-bit**. This value determines how many bits of data are used to represent each pixel's color value. A higher value allows for a greater range of colors, but uses more data and time to transfer.

Resolution

Resolution: **240 x 240**

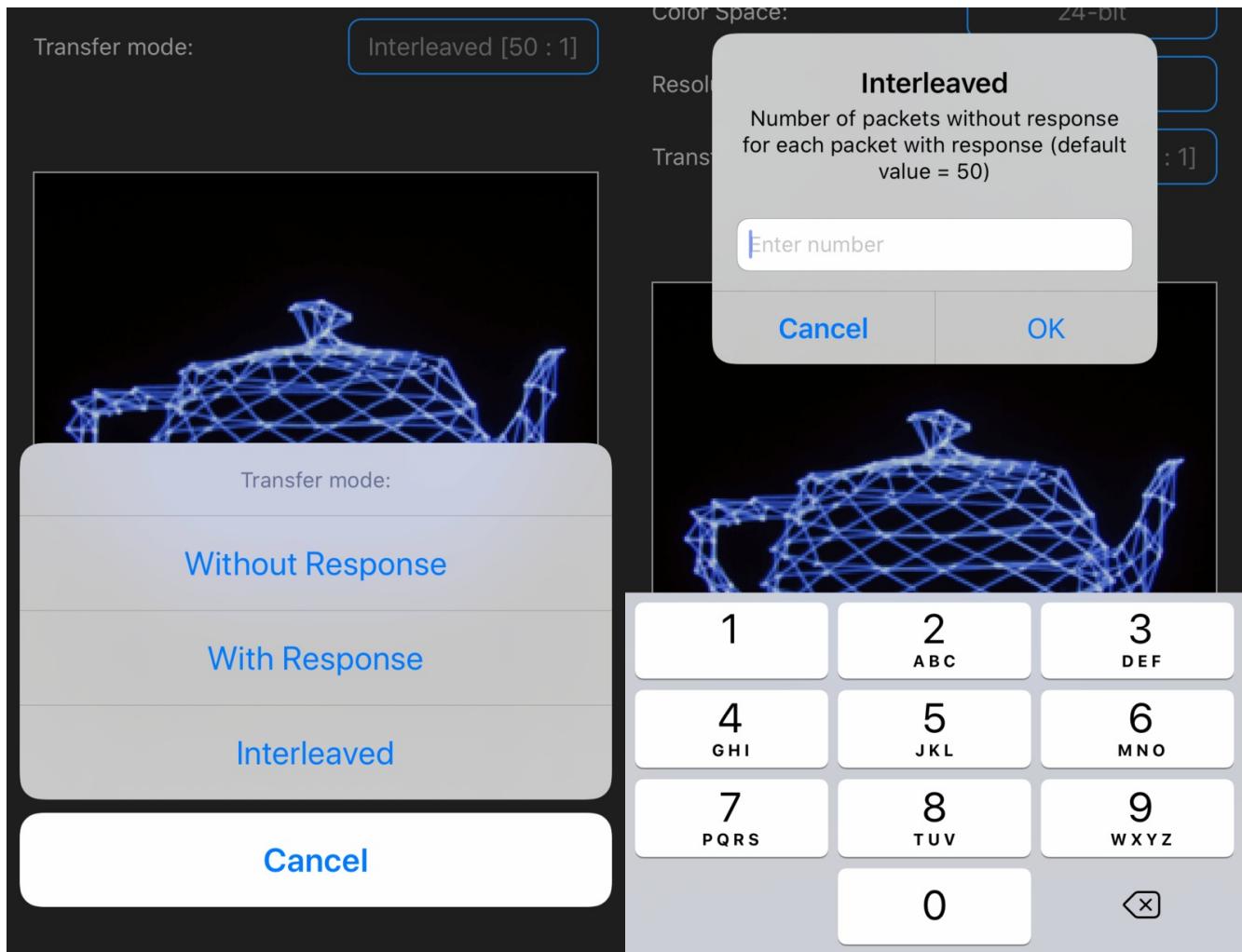


The Resolution setting allow you to choose from a list of preset image resolutions designed to fit common displays. Setting a resolution will update the selected image's pixellation as a preview.

Transfer Mode

Transfer mode:

Interleaved [50 : 1]

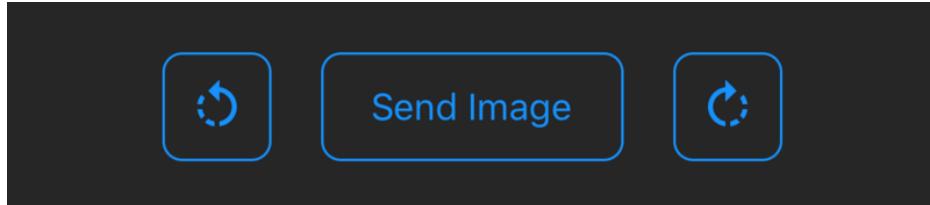


The Transfer Mode setting determines how often the app will require a **confirmation response** from the Bluefruit hardware before sending more data. Tapping the button offers three options:

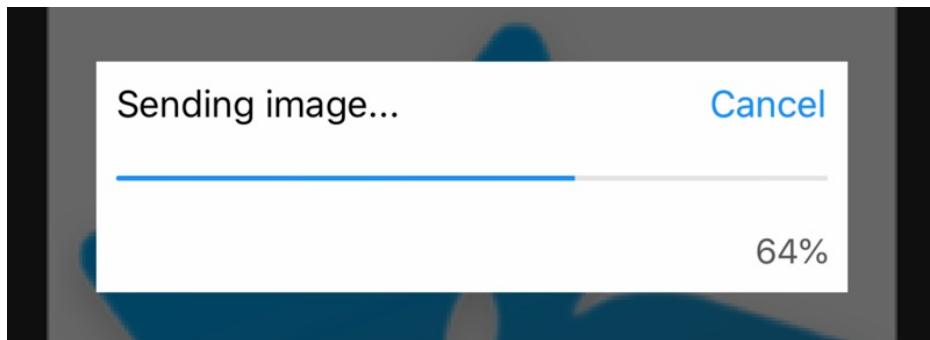
- **Without Response** (fastest, least reliable): App never waits for response before sending more data
- **With Response** (slowest, most reliable): App always waits for response to before sending more data
- **Interleaved** (customizable, varies): App sends a number of packets w/o response before sending one w response. The default value of 50 offers a nice middle ground between speed and reliability.

Example code written for Circuit Playground Bluefruit + TFT Gizmo is only compatible with default Transfer Mode of Interleaved [50:1]

Rotate & Send



The image can be rotated using the buttons on either side of the Send Image button.



Pressing **Send Image** will present a progress bar and begin sending data to the Bluefruit Hardware.

Pin I/O

This mode gives you a way to quickly control basic digital inputs and outputs. You are given a menu of all the pins that are available and you can set the direction and logic level. For some pins, you may also have PWM output & analog inputs! You can do quite a bit just with this method of control

Pin I/O	
DIGITAL	
Pin 3 Input	Low
Pin 4 Input	Low
Pin 5 Input	Low
Pin 6 Input	Low
Pin 7 Input	Low
Pin 8 Input	Low
ANALOG	
Pin A0 Input	Low
Pin A1 Input	Low
Pin A2 Input	Low
Pin A3 Input	Low

Pin I/O	
DIGITAL	
Pin 3 Output	Low
<input type="button" value="Input"/> <input type="button" value="Output"/> <input type="button" value="PWM"/>	
<input type="button" value="Low"/> <input type="button" value="High"/>	
Pin 4 Input	Low
Pin 5 Input	Low
Pin 6 Input	Low
Pin 7 Input	Low
Pin 8 Input	Low
ANALOG	
Pin A0 Input	Low
Pin A1 Input	Low

- Each row in the table represents a pin on your Arduino. Pin name and current state are displayed on the left side of the cell, while pin value is displayed on the right.
- Tap a row to change the relevant pin's current mode and value.
- Tap the row a second time to hide its controls

DIGITAL

Pin 3

PWM

113

Input

Output

PWM



Pin 4

Input

Low

Pin 5

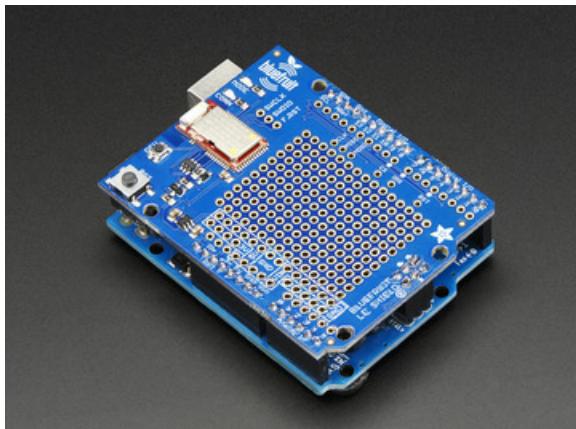
Read through the [Wiring](https://adafru.it/kCe) (<https://adafru.it/kCe>), [Configuration](https://adafru.it/kCf) (<https://adafru.it/kCf>) and [Usage](https://adafru.it/kCg) (<https://adafru.it/kCg>) pages for more details

Wiring Options

The Pin I/O mode lets you control an Arduino or compatible which is connected to a Bluefruit LE module.

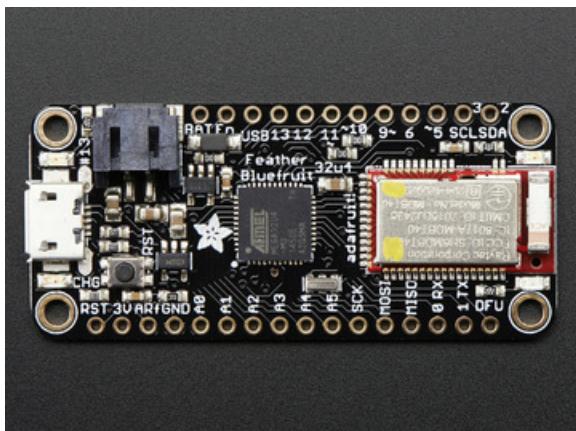
You can use the **nrf8001**-based or **nrf51822**-based breakouts. However, the setup will vary slightly.

Let's start with *wiring options*



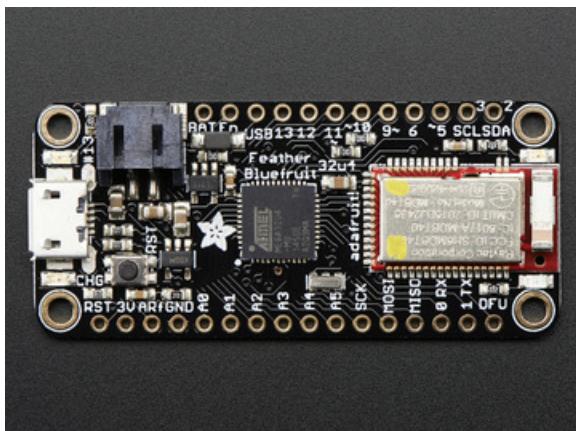
Arduino with Bluefruit LE Shield

If you are using the Bluefruit LE Shield then you have an **SPI-connected NRF51822** module. You can use this with **Atmega328** (Arduino Uno or compatible), **ATmega32u4** (Arduino Leonardo, compatible) or **ATSAMD21** (Arduino Zero). Your pinouts are **Hardware SPI, CS = 8, IRQ = 7, RST = 4**



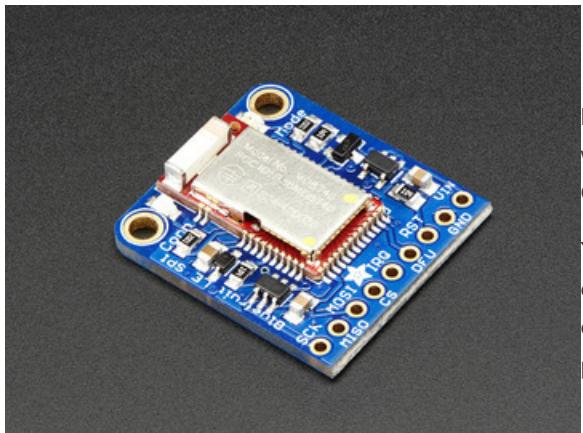
Bluefruit Micro or Feather 32u4 Bluefruit

If you have a Bluefruit Micro or Feather 32u4 Bluefruit LE then you have an **ATmega32u4** chip with **Hardware SPI, CS = 8, IRQ = 7, RST = 4**



Feather M0 Bluefruit LE

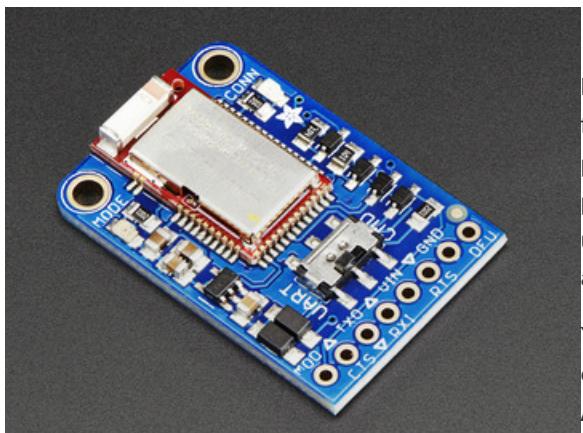
If you have a Feather M0 Bluefruit LE then you have an **ATSAMD21** chip with **Hardware SPI, CS = 8, IRQ = 7, RST = 4**



Bluefruit LE SPI Friend

If you have a stand-alone module, you have a bit of flexibility with wiring however we strongly recommend **Hardware SPI**, CS = **8**, IRQ = **7**, RST = **4**

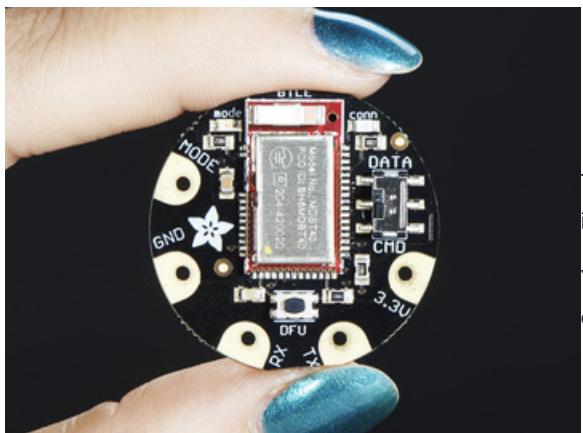
You can use this with **Atmega328** (Arduino UNO or compatible), **ATmega32u4** (Flora, Arduino Leonardo, compatible) or **ATSAMD21** (Arduino Zero) Your default pinouts should be **Hardware SPI, CS = 8, IRQ = 7, RST = 4**



Bluefruit LE UART Friend

If you have a stand-alone UART module you have some flexibility with wiring. However we suggest hardware UART if possible. You will definitely need to use the flow control CTS pin if you are not using hardware UART. You will need to set up the MODE pin as well since Firmata uses both command and data modes

You can use this with **Atmega328** (Arduino UNO or compatible), **ATmega32u4** (Arduino Leonardo, compatible) or **ATSAMD21** (Arduino Zero)



Flora BLE

This simplified UART friend is BLE for FLORA, its really only intended for use with Hardware Serial, there's no flow control.

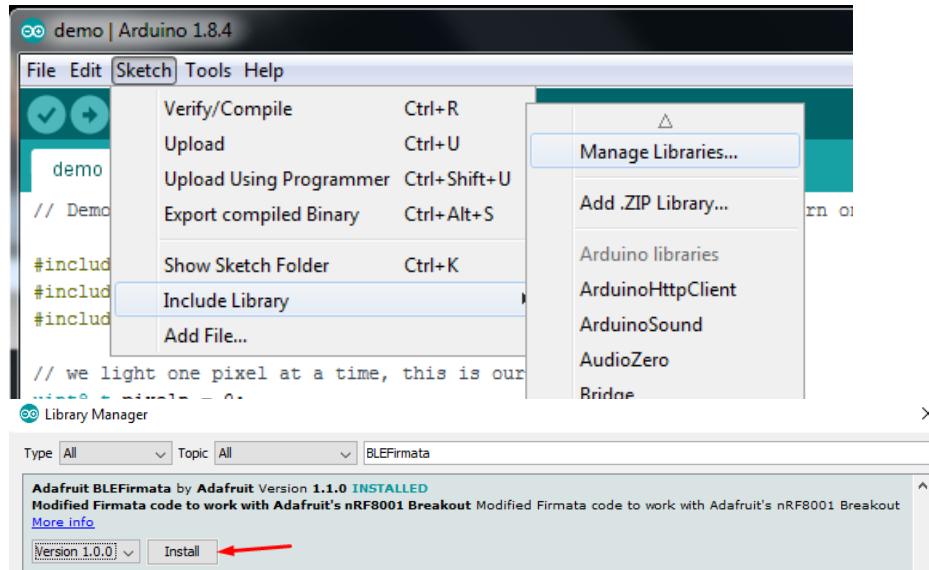
You can use this **ATmega32u4** (Flora) although in theory you could wire it up to a different processor

Library and Config Before loading Firmata BLE...

Make sure you have the basic tutorials working, that you can send/receive data via the UART with your wiring. If you can't talk to the module and send/receive data via the basic UART demo, Firmata won't work!

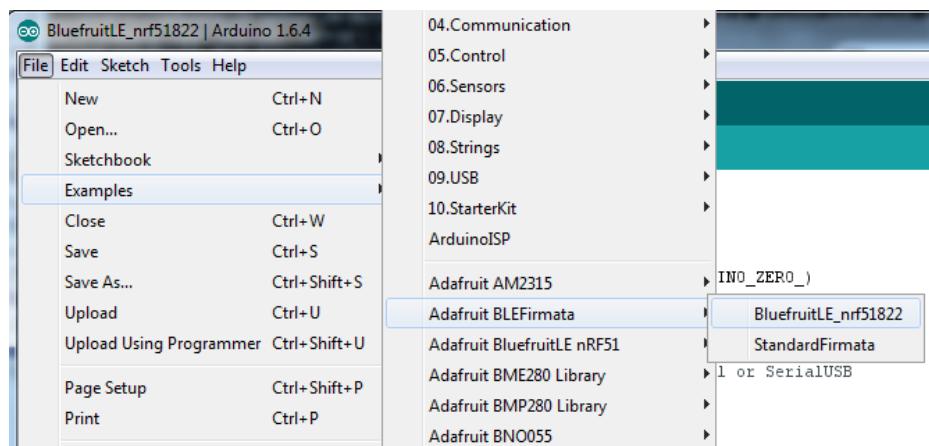
Install Libraries

You'll want to start by [installing the BLE Firmata library \(<https://adafru.it/fTO>\)](https://adafru.it/fTO). Use the Library manager to find Adafruit BLEFirmata and install it



Open Sketch and Configure

Restart the IDE and load the `Adafruit_BLE_PinIO->BluefruitLE_nrf51822` sketch



At the top of the sketch is the Firmata configuration section, there's also **Bluefruit LE pin configuration** in the `BluefruitConfig.h` tab

Bluefruit LE Config

Start by opening the `BluefruitConfig.h` tab. By default the sketch is set up for **hardware SPI** and **CS = 8**,

IRQ = 7 and **RST** = 4. This works by default for our Bluefruit Feathers.

You'll need to change your pins if you are using UART or different SPI pins.

Check out the generic "Configuration!" details for information on what every pin does. If you're using software serial, be sure to set up flow control and a MODE pin. (<https://adafru.it/kCT>)

```
// -----
#define BLUEFRUIT_UART_MODE_PIN      12 // Set to -1 if unused

// -----
// SHARED SPI SETTINGS
// -----
// The following macros declare the pins to use for HW and SW SPI communication.
// SCK, MISO and MOSI should be connected to the HW SPI pins on the Uno when
// using HW SPI. This should be used with nRF51822 based Bluefruit LE modules
// that use SPI (Bluefruit LE SPI Friend).
// -----
#define BLUEFRUIT_SPI_CS           8
#define BLUEFRUIT_SPI_IRQ          7
#define BLUEFRUIT_SPI_RST          4

// SOFTWARE SPI SETTINGS
// -----
// The following macros declare the pins to use for SW SPI communication.
// This should be used with nRF51822 based Bluefruit LE modules that use SPI
// (Bluefruit LE SPI Friend).
// -----
#define BLUEFRUIT_SPI_SCK          13
#define BLUEFRUIT_SPI_MISO         12
#define BLUEFRUIT_SPI_MOSI         11
```

Then in the main sketch, if you're not using hardware SPI, uncomment the connection style you want

```
// Create the bluefruit object, either software serial...uncomment these lines
/*
SoftwareSerial bluefruitSS = SoftwareSerial(BLUEFRUIT_SWUART_RXD_PIN, BLUEFRUIT_SWUART_TXD_PIN);

Adafruit_BluefruitLE_UART ble(bluefruitSS, BLUEFRUIT_UART_MODE_PIN,
                           BLUEFRUIT_UART_CTS_PIN, BLUEFRUIT_UART_RTS_PIN);
*/

/* ...or hardware serial, which does not need the RTS/CTS pins. Uncomment this line */
// Adafruit_BluefruitLE_UART ble(BLUEFRUIT_HWSERIAL_NAME, BLUEFRUIT_UART_MODE_PIN);

/* ...hardware SPI, using SCK/MOSI/MISO hardware SPI pins and then user selected CS/IRQ/RST */
Adafruit_BluefruitLE_SPI ble(BLUEFRUIT_SPI_CS, BLUEFRUIT_SPI_IRQ, BLUEFRUIT_SPI_RST);

/* ...software SPI, using SCK/MOSI/MISO user-defined SPI pins and then user selected CS/IRQ/RST */
//Adafruit_BluefruitLE_SPI ble(BLUEFRUIT_SPI_SCK, BLUEFRUIT_SPI_MISO,
//                           BLUEFRUIT_SPI_MOSI, BLUEFRUIT_SPI_CS,
//                           BLUEFRUIT_SPI_IRQ, BLUEFRUIT_SPI_RST);
```

Firmata Debug Config

Next up in the main sketch there's a few settings you'll have to configure. First up is debug and serial details

```
// Change this to whatever is the Serial console you want, either Serial or SerialUSB
#define FIRMATADEBUG Serial
// Pause for Serial console before beginning?
#define WAITFORSERIAL true
// Print all BLE interactions?
#define VERBOSE_MODE false
```

The first setting **FIRMATADEBUG** is how output is printed. 99% of the time you'll be happy with Serial, but for Arduino Zeros you may need to use SerialUSB for the native port.

WAITFORSERIAL determines whether the sketch waits for the Serial port to be opened before it runs. Set it to true while debugging/testing and open up the serial console to kick off the sketch. Set to false once it's working great

VERBOSE_MODE allows you to see all of the data passing between the BLE module and App.

Available Pins Config

Below the debug config is where you can set up what pins are available for the Pin IO app to twiddle

```
***** For Bluefruit Micro or Feather 32u4 Bluefruit *****
//uint8_t boards_digitaliopins[] = {0,1,2,3,5,6,9,10,11,12,13,A0,A1,A2,A3,A4,A5};

***** For UNO + nRF58122 SPI & shield *****
//uint8_t boards_digitaliopins[] = {2, 3, 5, 6, 9, 10, A0, A1, A2, A3, A4, A5};

***** For Bluefruit M0 Bluefruit *****
//uint8_t boards_digitaliopins[] = {0,1,5,6,9,10,11,12,13,20,21,A0,A1,A2,A3,A4,A5};
```

You'll need to uncomment *one* of these lines. Also, they are set up by default for the SPI Bluefruit module, on pins 4, 7, 8. For that reason, the hardware SPI pins and 4,7,8 don't appear. If you are using different pins you can re-add those to the list. If there are any other pins you don't want to show up in the app, remove those pins as well.

For the 32u4 and M0 examples, there's a lot of pins, & perhaps not all of them are necessary! You can have as few as you like.

Below the setup lines you can see the way we tell Firmata which pins do what

```
#if defined(__AVR_ATmega328P__)
  // Standard setup for UNO, no need to tweak
  uint8_t boards_analogiopins[] = {A0, A1, A2, A3, A4, A5}; // A0 == digital 14, etc
  uint8_t boards_pwmpins[] = {3, 5, 6, 9, 10, 11};
  uint8_t boards_servopins[] = {9, 10};
  uint8_t boards_i2cpins[] = {SDA, SCL};
```

Don't mess with these! Change only the digital IO pins array!

Upload and test

Once you're done compile and upload. Open up the serial console. You should see that the sketch was able to initialize the Bluefruit LE module, reset it and print out some details about the BLE firmware. It will now wait for the app to connect

```
Adafruit Bluefruit LE Firmata test
Total pins: 20
Analog pins: 6
Initialising the Bluefruit LE module: OK!
Performing a factory reset:
Requesting Bluefruit info:
-----
BLESPIFRIEND
nRF51822 QFACAA10
C20DDC1D1764441A
0.6.5
0.6.5
Jul 1 2015
S110 8.0.0, 0.2
-----
Setting name to BLE Firmata
```

The window has a 'Send' button at the top right. At the bottom, there are three dropdown menus: 'Autoscroll' (checked), 'No line ending', and '9600 baud'.

If you don't see anything, check:

- If you're using an ATSAMD21/M0 chip, do you have `FIRMATADEBUG` set to `SerialUSB`?
- Do you have `WAITFORSERIAL` true?

Once you know everything is working, set "#define WAITFORSERIAL true" to false so you don't have to open the serial console to have the Firmata code run!

Once you connect in Pin IO mode you can see a stream of commands that are received and acted upon

COM93 (Adafruit Feather M0 (Native USB Port))

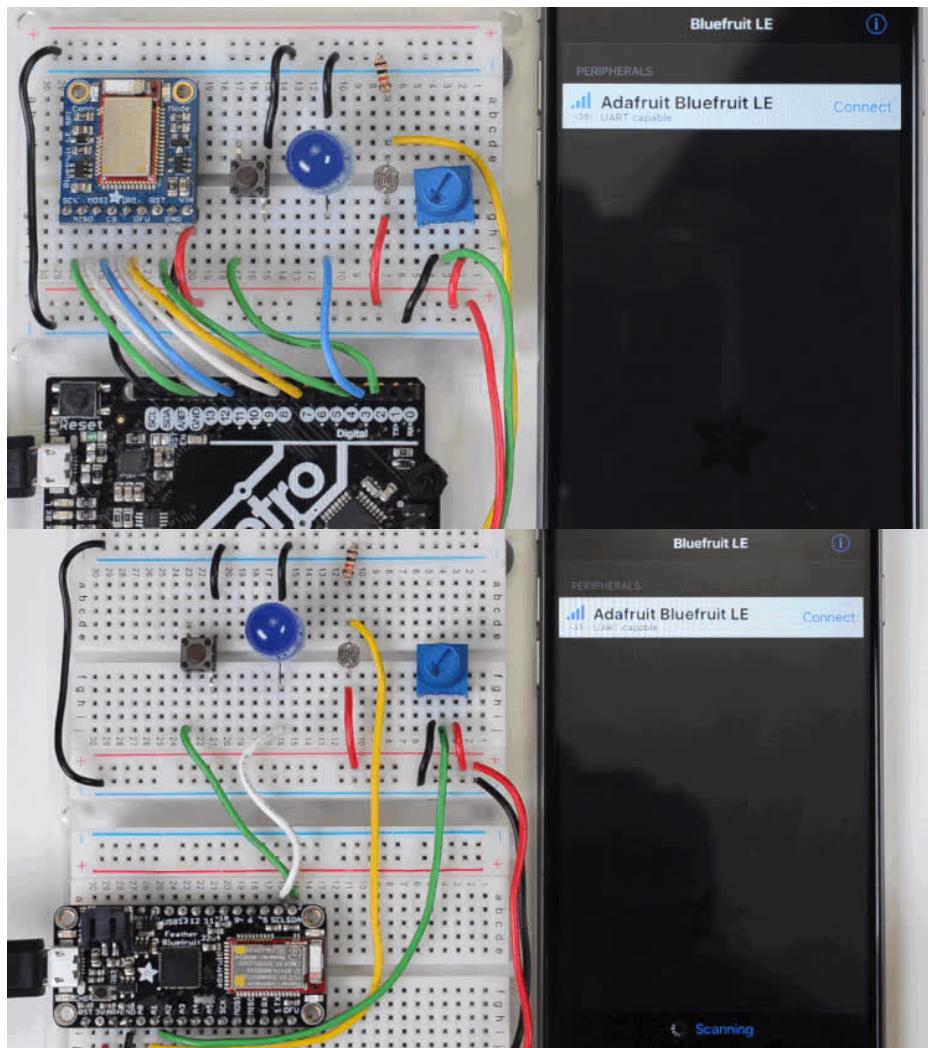
```
Set pin #21 to input
***** Sysex callback
***** Sysex callback
Analog mapping query
Set pin #0 to input
Set pin #1 to input
Set pin #5 to input
Set pin #6 to input
Stop reporting analog pin #7
Set pin #9 to input
Set pin #10 to input
Set pin #11 to input
Set pin #12 to input
Set pin #13 to input
Stop reporting analog pin #0
Set pin #14 to input
Stop reporting analog pin #1
Set pin #15 to input
Stop reporting analog pin #2
Set pin #16 to input
Stop reporting analog pin #3
Set pin #17 to input
Stop reporting analog pin #4
Set pin #18 to input
Stop reporting analog pin #5
Set pin #19 to input
Set pin #20 to input
Set pin #21 to input
Set pin #13 to output
Write digital port #1 = 0xFE mask = 0x20
Write digital port #1 = 0xDE mask = 0x20
```

Autoscroll No line ending 115200 baud

Usage

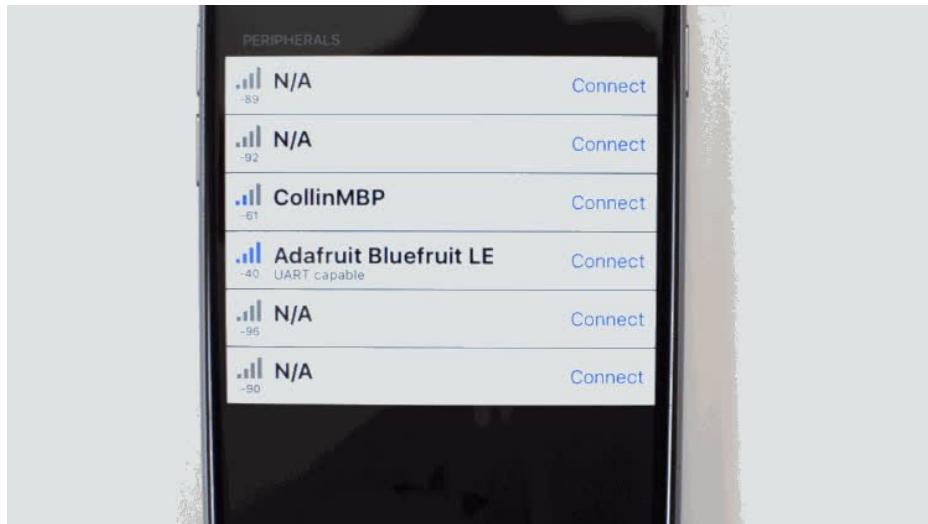
The Pin IO capability basically lets you control the pins of the Arduino one by one. Its meant for basic prototyping and control where you may not want to write a full app from scratch!

The app and sketch are also 'smart' in that when you connect, the app will query the Arduino what pins are available and what they can do!

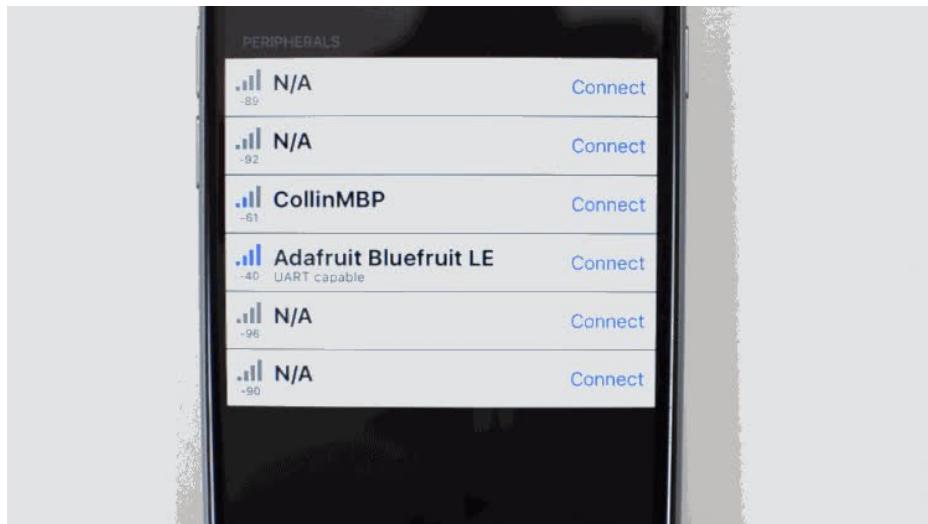


Initial Query

Make sure you have the most recent version of the app, and the correct configuration. During connection you'll see the app **Querying Pin Capabilities...** It will then get the correct details and fill out the pin map

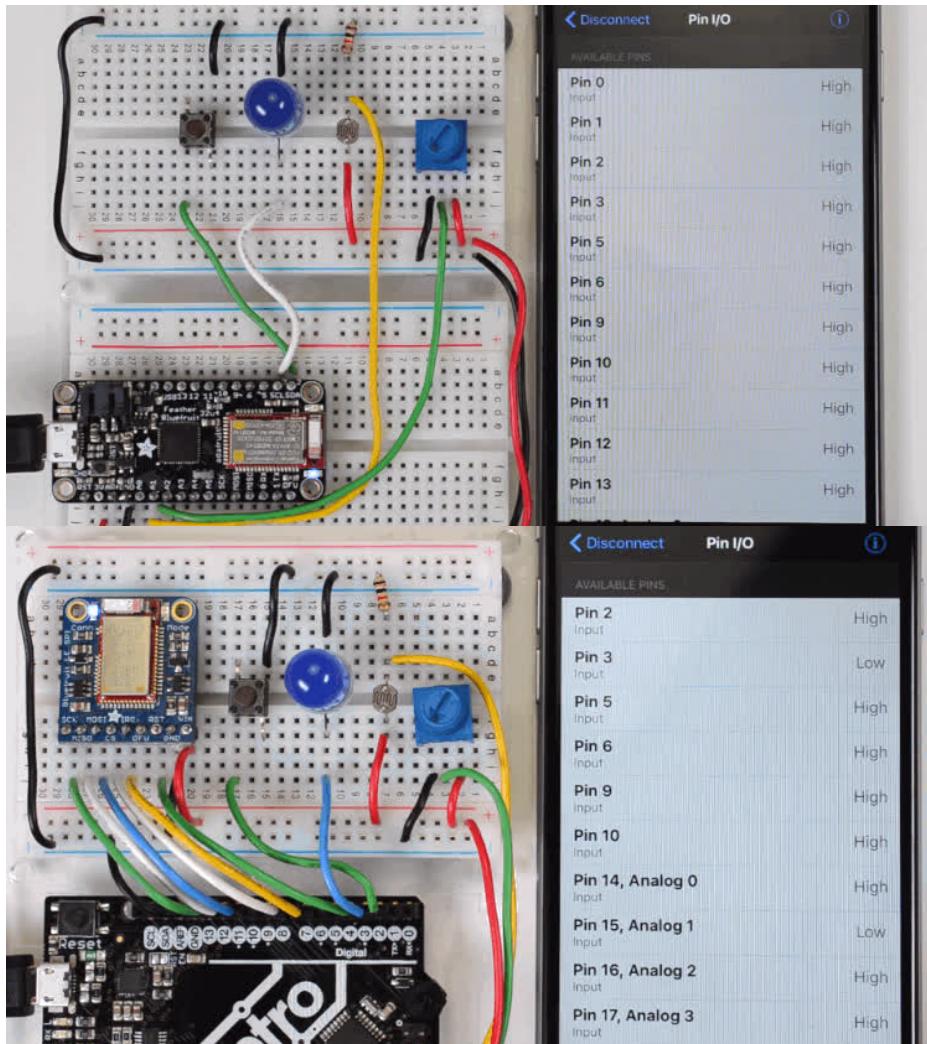


If the query fails, the app will default back to our old 'UNO' setup, which may not work well for you. Check debug output for the Bluefruit firmata sketch, make sure you're running the latest version of the sketch



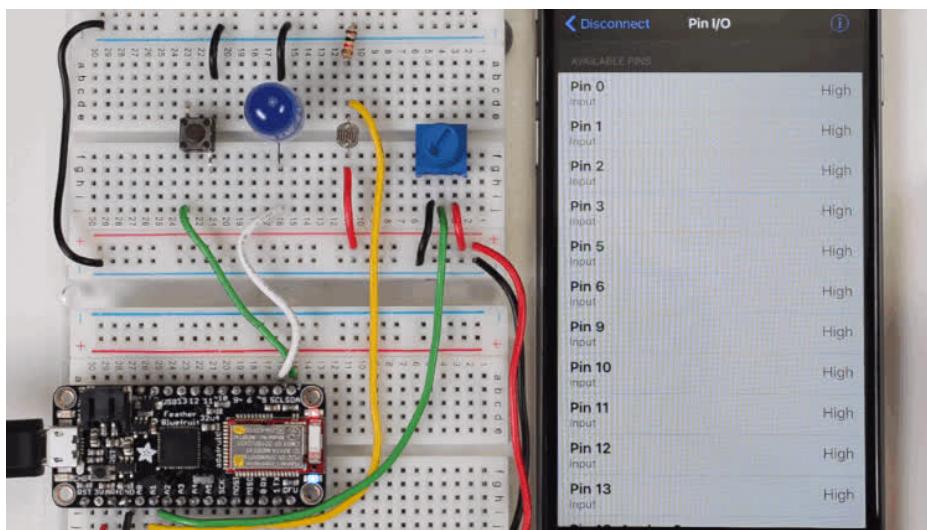
Digital Input

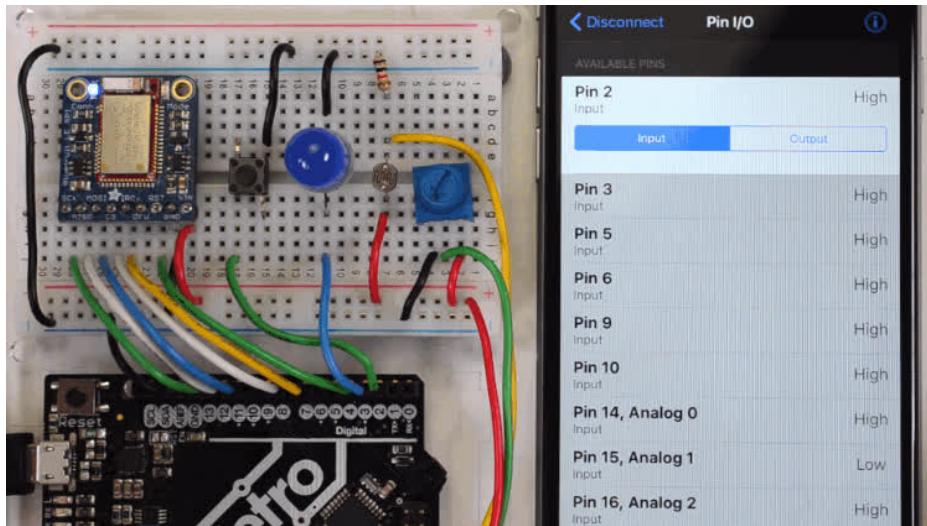
All of the pins default to digital inputs with built-in pullup resistors. This means that by default the pins read **HIGH**. When a button is wired to the pin & pressed or the pin is shorted to ground, you will receive a **LOW** signal



Digital Output

You can also set the pins to digital outputs. This will let you set the pins HIGH (3V or 5V, depending on the microcontroller voltage) or LOW (0V a.k.a ground). Great for turning on & off LEDs



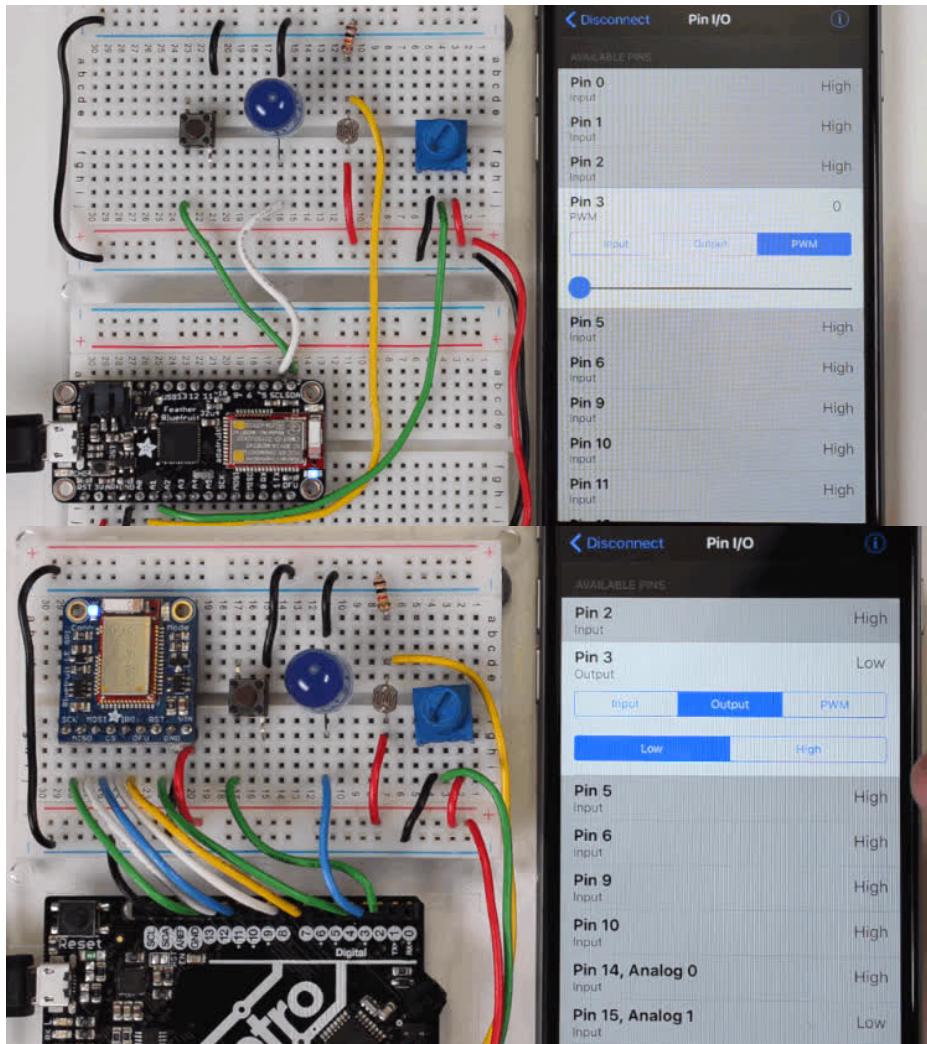


You can also wire it up to a PowerSwitch Tail (<http://adafru.it/268>) which gives you a safe way to control appliances!



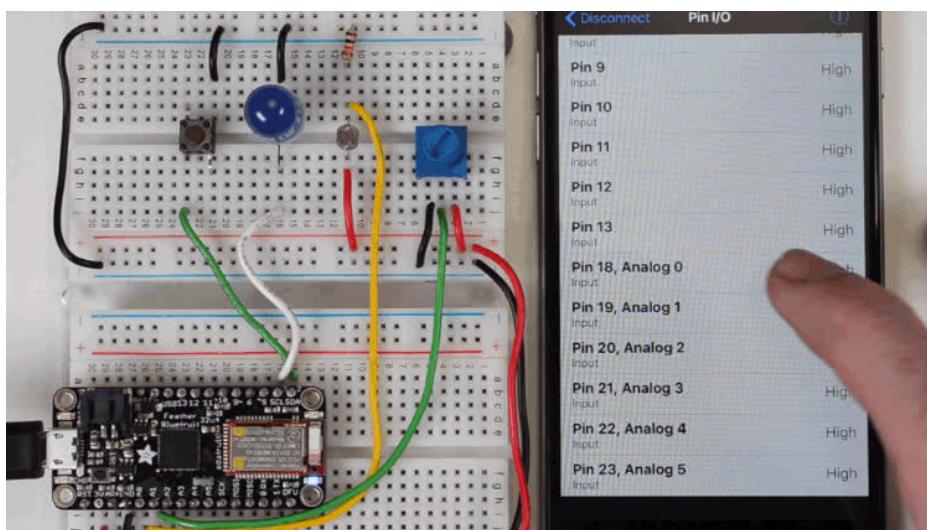
PWM Output

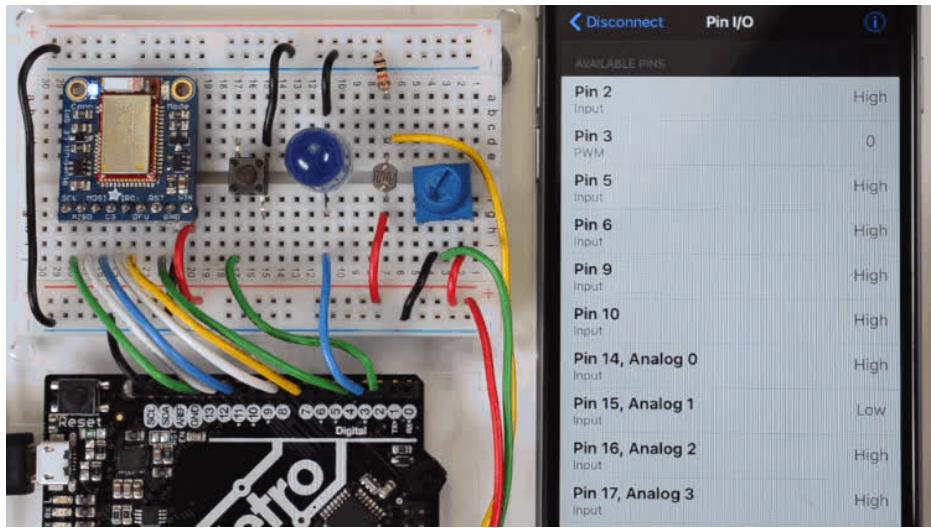
On some pins, there's PWM output available, you can dim or brighten an LED. You could also control a DC motor through a driver, if it can be PWM controlled



Analog Input

There are also some pins that have Analog pin capability. You can read various sensors or potentiometers that are wired to these pins. Its best to make sure you know how to get the analog sensor working first, using plain Arduino code, before wiring it up for use with Pin IO





Troubleshooting

If you're having trouble finding your Bluefruit device in the scanned peripherals list, it could be due to caching issues in the iOS or Android operating system. For a fix, try the following:

Cycle Bluetooth

Turn your mobile device's Bluetooth radio **off** and **on** again via your device's Settings app.

Cycle Power

Restart your mobile device by **powering it off** and **restarting**.

Clear Bluetooth Cache

On **iOS**, go to **Settings->Bluetooth**, tap the **i** button, and tap **forget device**. For **Android**, instructions can be found [here](https://adafru.it/F6x) (<https://adafru.it/F6x>).

